Smart Mobile Application for Public Transport Schedules - Logical Model

Georgi Krastev [1], Valentina Voinohovska [2]

[1] *University of Ruse "Angel Kanchev", Department of Computer system and technologies, 8 Studentska str., Ruse, Bulgaria*
[2] *University of Ruse "Angel Kanchev", Department of Informatics and information technologies, 8 Studentska str., Ruse, Bulgaria*

***Abstract* – Logical model of a portal for smart mobile application for public transport schedules on the base of three classes Model/View/Controller (MVC) has been described in this article. This concept based on design templates, leads to greater flexibility and possibility for repeated use in various applications.**

***Keywords* – logical model, Model/View/Controller, smart device, public transport, mobile web, data base.**

1. Introduction

Smart mobile applications are software applications that are designed to run on smart phones, tablets, and other mobile electronic devices. In this era of rapid technological advances, these applications have become one of the primary tools we use daily both in our personal and professional lives [10].

The developed application is a full featured website designed for smart devices with the following features:

- Provides public transport information for a specific location.
- Generates a route between two randomly selected stops for a given day and time of departure or time of arrival.
- Shows routes of certain lines as well as links to others.

DOI: 10.18421/TEM92-15
https://doi.org/10.18421/TEM92-15

Corresponding author: Valentina Voinohovska,
University of Ruse "Angel Kanchev", Department of Informatics and information technologies.
Email: voinohovska@ami.uni-ruse.bg

Received: 25 March 2020.
Revised: 23 April 2020.
Accepted: 29 April 2020.
Published: 27 May 2020.

Dijkstra's algorithm is used. It is also known as a single source shortest path algorithm. It is applied only on positive weights graphs. This algorithm is often used in routing [9].

Designing object-oriented software is difficult while designing object-oriented software for repeated use is even harder [3].

The architecture design MVC has been approved for creation of Internet sites [1],[2],[5]. The common form of the architecture is given in Fig. 1.

- **Model** – the application's kernel predetermined by the field it has been developed for. Usually these are data from real world that have been moulded and that has been worked on – input, change, visualization etc.
- **View** – visualization of data from the model. For example, view may consist of template classes. It depends on what graphic interface has been created and what platform has been used.
- **Controller** – additional class that takes data from the model or draw additional methods upon the model, processes data in advance and then give them up to the view. For example, a small object where transaction data are included may be created – as in controller: data for transaction from the model may be taken; dates from UNIX format may be set in a readable for the user format; currency may be transformed, for example, from dollars to euro; the round may be to the second digit instead data to be seen to the tenth digit as it is in the model and in the base. Creation of WEB graphic interface would lead to a very easy modification of HTML code – template will be considered as an ordinary HTML page.

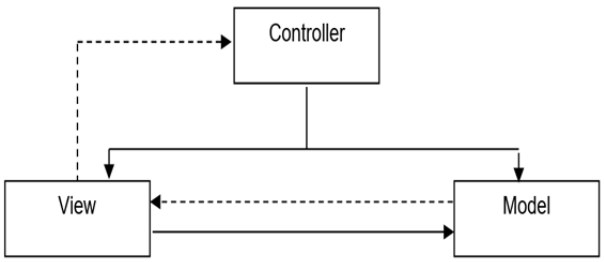

Figure 1. Architecture of MCV

2. Model of the System

An application implementing the MVC pattern consists of three modules: model, view and controller [4], [6]. The Model/View/Controller design pattern is very useful for architecting interactive software systems [8].

MCV has been used as template for architecture design of the application. It is necessary every object to support a function for error removal.

- **Base object** – other base functionalities that would be accessible from every object are necessary, for example, data of the configuration file. That is why all objects inherit one main base object – BaseClass.
- **CDB class** – functions for work with data base management system are separated in another object. The relevant features are kept there – connection creation, order fulfillment etc.
- **Data model** – CModel class is created here. Its purpose is to manage properly the tables in data bases. The relevant relations and orders are accomplished here according to data organization.
- **Functions for file presentation** – they are necessary to the proper arrangement of data on the display, choice of correct templates, and type of smart mobile device. Correct presentation of pages for the relevant type of platform is necessary to be done; that may be xHTML and i-mode (HTML). Another main functionality of this class is internationalization – the language that will be used for a definite message to be written is determined here. The name of the class is CView.
- **Templates system** – particular templates as well as their analysis are used here. The name of this class is CParser.

All pages visualizations have definite specifications and necessary number of functionalities. Every definite part of the application, concerning the logic of data processing and presentation will use separate controller – a separate object [12]. For example, functionality of the site for generating itinerary between two stations will have a separate object from functionality for scrutinizing the transport routes. All these objects are controllers. They have a lot of common qualities. That is why base controller is created that is inherited by all the rest. The name of the class of the main controller is CController.

The logic model of the system is given in Fig. 2. Every object inherits the base class and acquires abilities to find defects and access to configuration file. The class that manages DBSM (data base system management) is CDB and its representative (object) can be found only in CModel class.

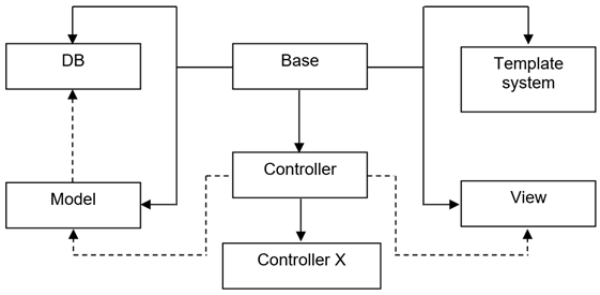

Figure 2. Logic model of the program system

To access data base all objects use CModel class - that guarantees data completeness. All orders to data base are defined in the model. The model gives the relevant functions for data obtaining. Tropology of data base remains hidden to the rest of the objects. The model has at its disposal ready functions for initial installation and help for renovation fulfillment.

A few more classes and relevant objects are necessary to add the functionality for fulfillment of analysis and administration of the remote site. Administration has been fulfilled as a separate controller from the management controller of the user's part of the program. The search of itinerary between two stations has been accomplished as additional class of the model. The detailed model is given in Fig. 3.

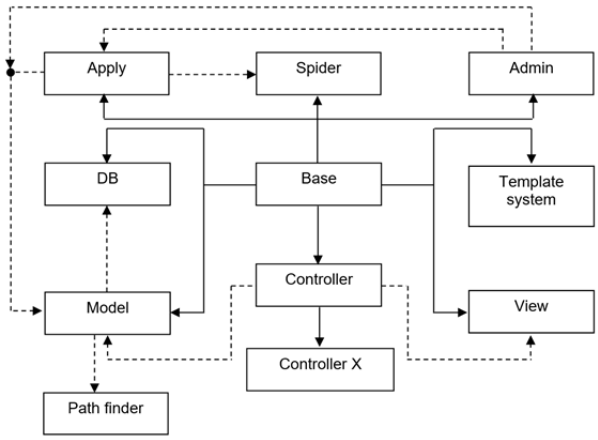

Figure 3. Detailed model with modules for processing the remote site

- CSpider – it cares for visiting and analysis of the external site. The presented synthesized and arranged results are ready for further processing. The main functionality of this class is to carry out the connection with the remote site, to prepare the communication in appropriate way so that the desired pages to be opened. After that an interpretation of result and preparation of information suitable for work is fulfilled.
- CApply – it arranges and synthesizes the derived information, uses the methods of the model to input the information in tables designed to update

the data base. Functions that work by stages are included here: extraction of list with stations, transport routes and their input in data bases; extraction of total information about a definite transport route – itineraries, stations, and schedule for each station, working days or holidays.

- CAdmin - controller of administrative panel. Functionalities necessary to install and administrate the application are developed here.
- CPath_Finder – additional class to the model. It serves to find an itinerary between two stations. The process of finding is complex and that is why it is realized as a separate class.

The main model of the site is an independent and fully functional WEB site for mobile phone that may be applied without any modifications for every town or out-of-town transport system. The detailed model includes the main one with possibilities to extract data for public transport in the capital and its particular arrangement in data base. If the application is developed for another environment, only classes from the extension need to be modified. These classes cannot be universal, that is why the logical model is divided into two parts – the main one that is the kernel of the application and the extended one – it gives resolution of a particular case in a definite environment.

Because there is an automated analysis of external WEB site, some of the pages may be interpreted incorrectly. Different factors as change of form for output of the relevant pages, errors in the remote site (that leads to partial or incorrect information output), lack of communication with the remote site and others may cause these errors. That is why functionality is necessary to be created that will protect from these errors that may cause damage or incorrect work of the mobile WEB site. Duplication of tables has been created for that purpose – a set to output the information to the final user and another that is used to upgrade the information. If the renovation has been done successfully, the information from the renovated tables has to be accessible to final users. Data transferring between tables is not a decision because it takes too long time and the site is possible to be accessible at the time of transferring that would lead to incorrect work of the application. For that reason, the site has to use the tables that had served for renovation when information is published. Then 'transferring' is instantaneous and there is no danger to stop the application. After that the tables having been used to present the application are deleted. At next renovation new tables are created and the cycle continues. The cycle of renovation of information from initial establishment is presented in Fig. 4.

At the initial establishment of application, there are no data. Renovation is necessary to be done to fill up the information (renovation 1). After that, it is possible for the information to be published - step 2. Step 3 – renovation 2 has been published. During the process of renovation, the tables that had contained information from renovation 1 delete themselves automatically. Two sets of tables are always available in data base as the unnecessary ones delete themselves. The model has the possibility to delete the unpublished tables (only those that are used from the final user remain).

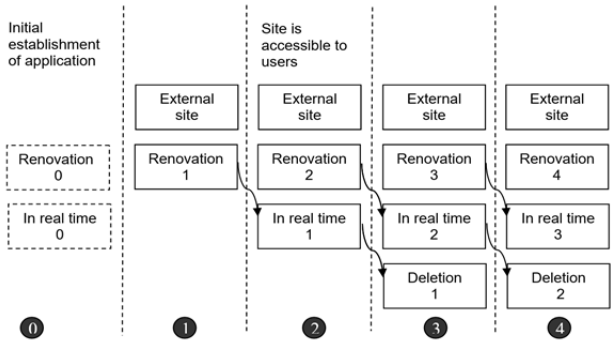

Figure 4. Cycle of renovation of information

Site must not stop in the process of working. The model possesses functions that fulfilled the above mentioned operations automatically and safely without making any errors. In a variable registered in data base, there is information about which set of tables is used to visualize the application in front of users in real time.

3. Main Functionalities

The described application is portal of smart devices for public transport schedules. It works in two modes: user mode and site administration. The administrative panel serves for information input and renovation in data bases while the user interface serves for information output [7], [11], [13].

The separate modules concerning the information management are presented in Fig. 5.

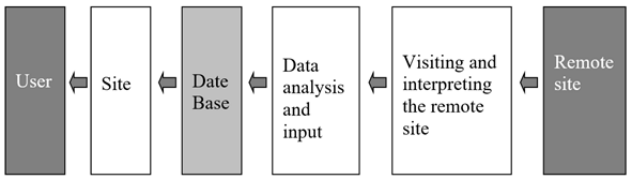

Figure 5. Information stream

4. Data Organization

The preserved data are as follows:

- Bus stops list.
- Transport routes list (buses, trolleys etc.).
- Itinerary for every transport route – a list with bus stops where every transport route passes to a definite direction.
- Transport routes schedule for each bus stop on working days and at weekends.
- Temporary preservation of generated itinerary.

Some of the data base tables will be doubled as the application will derive data by visiting another web site. General mode of the tables and connections between them is given in Fig. 6.

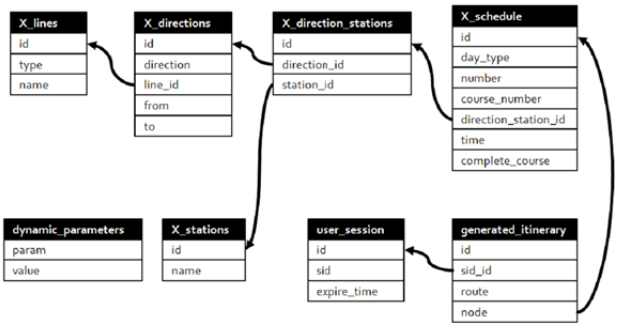

Figure 6. Data base model

'X' symbol in front of the name defines those tables that are used twice to be presented to the final users and renovate the information. Detailed description of each table is given below:

1) X_lines table

It contains information for transport routes. It has the following fields: id INT (11) – primary key; type_id INT (11) – transport type (bus, trolley, tramway etc.); name VARCHAR (225) – key that is used only when visiting the site that is source of information.

2) X_stations table

It contains information about stations' names: id INT (11) – primary key; name VARCHAR (225) – station's name.

3) X_directions table

It contains information about directions of all transport routes. Generally every transport route has two directions – to one direction and backwards. There are transport routes that have more than two directions. The table is as follows: id INT (11) – primary key; direction VARCHAR (225) – direction number; line_id INT (11) – transport route identifying key; from VARCHAR (225) – name of place from where the transport heads to that direction; to VARCHAR (225) – name of the place where the transport reaches that direction.

4) X_direction_stations table

It contains list with stations where transport passes to a definite direction: id INT (11) – primary key – it is a definite station for a definite direction of a particular transport route; direction_id INT (11) – direction identification; station_id INT (11) – station identification.

5) X_schedule table

It contains schedule of every transport when passing all station: id INT (11) – primary key; day_type INT (1) – whether it is working day, pre-holiday or holiday; number INT (11) – serial number of the station in the course; course_number INT (11) – serial number of the course; direction_station_id INT (11) – station identification in a particular direction; time INT (11) – time of passing by that station; complete_course INT (1) – it specifies whether this is full course or it serves only some of the stations.

6) dynamic_parameters table

This table contains particular parameters for the application necessary to its work and renovation. These parameters are not in the configuration file for their automatic changes are necessary during application work: parameter VARCHAR (225) – key or name of parameter; value VARCHAR (225) – parameter value.

Parameters that serve to install the application are as follows:

- live_db_prefix = 0 – prefix of tables used for the application (what users see) – when changing this parameter the application starts using the renovated tables.
- update_db_prefix = 0 – prefix of tables used to update the information by visiting an outside site.
- test_key = 'ctmi' – key that helps to test the update information before being published for the final user.
- transport_tram = 1 – identification of type of transport – tramway.
- transport_trolley = 2 – trolley; transport_bus = 3 – bus.
- transport_metro = 4 – metro direction_day = 1 – identification of working day.
- direction_preholiday = 2 – pre-holiday; direction_holiday = 3 – holiday.

7) user_session table

This table is used to save the user's session. This is necessary when there is transmission of great amount of information between two stations:

- id INT (11) – primary key.
- sid VARCHAR (225) – session key transmitted among the stations.

- expire_time INT (11) – date and hour, when session expires after having been inactive. Inactive sessions are erased to prevent the unnecessary loading of information and work delay of application.

8) Generated_itinerary table

The information of generated itineraries is kept here. When finding an itinerary between two stations there is an excerpt of information of the generated itinerary for the user. If the user goes on, visualized details of the whole generated path are necessary to be given. That is why the creation of table for temporary preservation of information of generated itinerary is necessary. The main purpose of the table is to save productiveness since without its use the same itinerary will be generated again:

- id INT (11) – primary key.
- sid_id int (11) – identification of the used session.
- route int (11) – identification of generated itinerary. It is necessary if there is more than one itinerary to distinguish between separate paths.
- node int (11) – identification of a definite station at set time for transport route in definite direction. This field shows the elements of generated itinerary.

5. Conclusion

Smart mobile applications are software applications that are designed to run on smart phones, tablets, and other mobile electronic devices. In this era of rapid technological advances, these applications have become one of the primary tools we use daily both in our personal and professional lives.

Logical model of portal for smart mobile application for public transport schedules has been described in this paper. The article has been presented on the base of Model-View-Controller or MVC – architectural template for design. Each template focuses on a definite object-oriented design problem. It describes when it is applied, whether there is possibility to be applied in the presence of other restrictions in the design as well as its results and defects.

Acknowledgements

This publication is developed with the support of Project BG05M2OP001-1.001-0004 UNITe, funded by the Operational Programme "Science and Education for Smart Growth", co-funded by the European Union trough the European Structural and Investment Funds.

References

[1]. Bansiya, J., & Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, *28*(1), 4-17.

[2]. Chiarelli, A. (2016). *Mastering JavaScript Object-Oriented Programming*. Packt Publishing Ltd.

[3]. David P. (2020). *Guide to Efficient Software Design An MVC Approach to Concepts, Structures, and Models*. Springer.

[4]. Deck, P. (2016). *Spring MVC: a tutorial*. Brainy Software Inc.

[5] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1993, July). Design patterns: Abstraction and reuse of object-oriented design. In *European Conference on Object-Oriented Programming* (pp. 406-431). Springer, Berlin, Heidelberg.

[6]. Ganeshan, A. (2016). *Spring MVC Beginners Guide*. Packt Publishing.

[7]. Huk, M., Maleszka, M., & Szczerbicki, E. (Eds.). (2019). *Intelligent Information and Database Systems: Recent Developments* (Vol. 830). Springer.

[8]. Leff, A., & Rayfield, J. (2001). Web-application development using the Model/View/Controller design pattern. *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference*, Seattle, WA, USA, 118-127.

[9]. Singal, P., & Chhillar, R. S. (2014). Dijkstra Shortest Path Algorithm using Global Positioning System. *International Journal of Computer Applications*, *101*(6), 12-18.

[10]. Siuhi, S., & Mwakalonge, J. (2016). Opportunities and challenges of smart mobile applications in transportation. *Journal of traffic and transportation engineering (english edition)*, *3*(6), 582-592.

[11]. Silberschatz, A., Korth, H. F., & Sudarshan, S. (1997). *Database system concepts* (Vol. 5). New York: McGraw-Hill.

[12]. Wasserman, A. (2010). Software engineering issues for mobile application development. *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*, 397-400.

[13]. Younas, M., & Awan, I. (2016). *Mobile Web and Intelligent Information Systems*. Springer International Publishing.