Research on the release management technology of the mobile application for electricity dispatching

Jinxia Dai1*, Donghai Li2, Baoshan Gao3, Ling Zhou4 and Wenjing Chen5
1 Electronic Information Engineering, Nanjing University of Information Science and Technology, Nanjing, Jiangsu, 211106, China
2 Nanjing Institute of Technology, Electrical engineering and its automation, Nanjing, Jiangsu, 211106, China
3 School of Control and Computer Engineering, North China Electric Power University, Beijing, Beijing, 102206, China
4 Geographic Information System, Hohai University, Nanjing, Jiangsu, 211106, China
5 Computer Science and Technology, Hohai University, Nanjing, Jiangsu, 211106, China
*Corresponding author’s e-mail: daijinxia@sgepri.sgcc.com.cn

Abstract. With the continuous development of the power system and the rapid growth of the scale of the power grid, the information that needs to be monitored, inquired, and maintained in the power grid dispatching system is increasing. The traditional way of understanding the real-time operation status of the power grid through dispatching workstations is increasingly unable to meet the requirements of modern power grids for monitoring, operation and maintenance in real-time, timeliness, and anywhere. In the face of such a huge business demand market, how to develop mobile applications that meet customer needs with minimal investment in resources and efficient development efficiency has become our research direction. This paper discusses a release management technology of the mobile application for electricity dispatching that meets the above requirements. This technology adopts the design mode of "shared application + personalized page", without too much intervention by developers, Users can edit and compress their own personalized pages, then upload to the version server and update the main page on the mobile devices themselves; this technology improves the efficiency of opening graphics file and the response time of operation through caching the graphic files and only refreshing the operation area mechanisms; through the publish/subscribe message bus mechanism, the hot update effect of the main page is realized; the service is restored through one-click, which provides users with more operating space.

1. Introduction
With the continuous development and expansion of the electricity dispatching control system business, the traditional way of understanding the real-time operation status of the power grid through the dispatching workstation is increasingly unable to meet the requirements of modern power grid control for monitoring, operation and maintenance of real-time, timeliness, and anywhere. In this context, how to develop mobile applications that meet customer needs with minimal investment in resources and efficient development efficiency has become our research direction.
In order to achieve this efficient mobile application development model that meets the needs of power grid regulation business, we got rid of the traditional "one region, one application" development idea, and researched and realized the combined architecture model of "shared application + personalized page". In the traditional one regionone application model, we need to develop independent mobile applications for each customer. This is a model with high manpower requirements, high maintenance costs, and low development efficiency; the current "shared application + personalized page" development architecture extracts the common commonalities in business requirements and deposits them into the underlying mobile applications; separates diversified personality points, and fulfills this differentiated personality needs through the version release interface update method. Using this method of combining commonality and individuality not only reduces our investment in development resources, but also greatly improves our development efficiency. In this new architecture model of "shared application + personalized page", the performance and functions of the mobile application is particularly important. The time of drawing, the hot update, the update and restoration of the version, and more Humanized operation etc. These points are all needed to be considered in the mobile application[1].

2. Overall Framework

The mobile application for electricity dispatching adopts a new "shared application + personalized page" combined architecture design model. With mobile applications as the platform support, access to services through a secure access mechanism to obtain unique and personalized pages in each district. The overall framework is shown in the figure below[4].

![Figure 1. Common application + personalized interface framework.](image1)

The mobile application that adopts this "shared application + personalized page" design model, when there is a new customer demand, without too much intervention from the developer, the user can quickly edit out their own main page by using the mobile application editor provided by us and then upload it to the server through the version release platform. The mobile application terminal provides version download and automatic refresh functions to achieve a hot update effect. This is the realization idea of common mobile application with different pages and different functions for electricity dispatching. The schematic diagram of release update is as follows.

![Figure 2. Auto update framework.](image2)
3. Technical Difficulties

3.1. Configurable multi-page processing technology

In view of the wide range of user needs for mobile application display, monitoring, query and other business types, the traditional single-page integration model has gradually been unable to meet the needs of users, and switchable multi-page model have become a new demand trend. For this reason, the mobile application for electricity dispatching divide the main active screen into Fragment and Tab sections. Fragment is a common container for multiple pages, and its loaded content changes with the switching of tabs, and each tab corresponds to a different page entry. The design diagram of the main activity screen of the application is shown below[6].

![Design of the main activity interface.](image)

Figure 3. Design of the main activity interface.

In order to realize the different requirements of the main activity entrance interface of each application in each province and district, the configurability technology of the main page is introduced in the design. By introducing a configuration file with Json format, the number of pages, display positions, icon references, text labels, response to events and other attributes of the multi-page are formulated. The configuration file attribute conventions are as follows.

| Attribute    | Function                        |
|--------------|---------------------------------|
| index        | Controls the order of pages.    |
| title        | Title for the page.             |
| normalicon   | Icon for the normal table.      |
| activeicon   | Icon for the activated table.   |
| clickref     | The response to the clicked-event. |

Considering that mobile devices are limited by the screen size, the current agreement is formulated to support the coexistence of up to 5 tab pages. The json configuration file is a bridge to achieve multiple pages effects. After the user edits the page using the mobile application editor, the configuration file needs to be generated according to the configuration file protocol rules, and the page files are compressed and uploaded to the file version management server. After the mobile application client downloads the compressed file and stores it in the external storage space of the application, it
first decompresses the compressed package, and then queries whether the json configuration file is contained in the folder. If the configuration file is not found, a prompt is given, and the application will not do anything; if the queried configuration file exists, the mobile application will read, parse and encapsulate the file. After completion, the tab area of the main activity page will reference the new resource file to fill the page according to the value of the configuration file, and set a new response event at the same time; the fraction area of the main activity is a shared display container for multiple pages, and the first is loaded by default. The page configuration and repaint is as follows.

![Diagram](image.png)

Figure 4. Schematic diagram of interface configuration redraw.

3.2. Graphics file caching strategy and update technology

Due to the limitation of the mobile device's own processing capabilities, frequent network access and data file downloads will affect the product experience. Each time the traditional electricity dispatching mobile application opens a picture, it must access the server through Ajax request to query and obtain the graphic file, and then the mobile terminal draws and displays the canvas according to the specifications of the graphic file. This way, each time a drawing is opened, the graphics file is obtained through the server. There is an obvious waiting process from request to display. When encountering large graphics files, this waiting process appears to be longer.

In order to shorten the opening time and improve the user's experience except the first time, the graphic file silent cache technology and database file management scheme are imported. When drawing a picture for the first time, through an Ajax request, the graphic file is obtained from the server and drawn in the foreground. In order to prevent the page display that blocks the main thread or the interface freezes and other bad experience effects, a multi-threaded task mechanism is imported to download the graphic file and save. Traditionally, a task is processed and a thread is created. When the task processing is completed, the current thread is destroyed. With the continuous influx of tasks, there will be situations where tasks are continuously created and task threads are destroyed. Every task creation and destruction needs to acquire and occupy system processing resources and memory storage resources, which affects the operating efficiency of the system and applications. The introduction of thread pool management technology here can well avoid the waste of resources caused by frequent creation and destruction of task objects. The thread pool is a collection of threads. When the system starts, some idle threads will be created. The program will pass a task to the thread pool, and the thread pool will start a thread to perform this task. After the execution is over, the thread will not Die, but return to the thread pool to become idle again, waiting for the next task to be executed. The thread pool mainly includes four basic components: thread pool manager, worker thread, task interface, and task queue. Thread pool manager: used to create and manage thread pools, including creating thread pools, destroying thread pools, and adding new tasks; Worker thread: thread in the thread pool, in a waiting state when there is no task, and can execute tasks in a loop; Task interface: the interface that each character must implement for the work thread to call the execution of the task. It mainly stipulates the entry of the task, the finishing work after the task is executed, the execution status of the task, etc. Task queue: used to store unprocessed tasks and provide a buffer mechanism[2-3].
In the mobile application for electricity dispatching, this thread pool task mechanism is used to locally cache the image file and write the corresponding file storage path, file name, file version and other basic information into the database. When accessing the graphic file again, check whether the graphic file has been cached through the database query. If it has been cached, the graphic file of the specified path will be read for page drawing, effectively avoiding the network access and downloading time process and shortening the drawing time of the graphic file is determined[7].

3.3. Hot update and one-key recovery technology

In mobile applications, each activity runs in the main thread, which is mainly used for interface display, user interaction and simple business processing. Any time-consuming operation must be processed in other threads, otherwise it will easily cause people to be unable to respond properly. This needs to create communication requirements between the main thread and other threads, and between other threads. Traditionally, the handler mechanism is used to communicate between threads. This mechanism has a high degree of coupling between event publishers and event receivers, a large amount of code and poor module independence.

The message publish/subscribe bus mechanism is selected in the mobile application for electricity dispatching to achieve the hot update effect of the interface. For this reason, the handler mechanism is abandoned in the new technology, and the publish/subscribe message bus mechanism is introduced, which simplifies the communication between the components in the application and between the components and the background thread, effectively separating the event sender and the receiver, making the function realization become Concise and clear. This publish/subscribe message bus has three main components: events, event subscribers, and event publishers, which are independent of each other, avoiding complex and error-prone dependencies and life cycle issues. The framework implementation model mainly includes: custom events, register events, cancel events, send events, and process events[5].

The hot update technology of the mobile application for electricity dispatching control registers an event when the main activity window is started, send the event in the version management interface, and the event subscriber in the main activity window will start the page update operation after receiving the event. Upon exiting the active window, the subscription for the event will be cancelled.

After the interface version of the traditional mobile application is updated, if there is a problem, the entire page version release and update process can only be repeated. For this reason, it is particularly important to introduce the default built-in version one-key recovery technology. One-key recovery technology, when the updated new interface does not meet the needs of users, through one-key recovery technology, the mobile application can directly read, parse, and package the built-in configuration file, load the built-in interface file, and realize the one-click recovery effect of the interface version.

4. User feedback and test results

Through the actual use and feedback of multiple sites, it is concluded that after using the configurable model of common application and personalized page, when facing different needs of different locations, it not only greatly reduces the developer’s human resource input, but also improves the product’s performance. And it also allows users more autonomy and good controllability.

Through multiple rounds of self-test data comparison and the acceptance test of the Jiangsu Software Product Acceptance Center, after the graphic file is cached by the thread pool strategy, when a screen is opened, only the first time is relatively time-consuming. When this picture is opened, dynamic data, coloring information and other related information will be directly obtained from the local cache to draw and display the graphics on the canvas. The display time will be significantly shortened, which greatly improves the user experience. The following data are tested in Huawei tablet BTV-DL09 device, memory 4.0G, Android 7.0, resolution 1600x2560.
Table 2. AttributeTest Result

| ID | Test File   | File Size (KB) | Open times(ms) | Cached times(ms) |
|----|-------------|----------------|----------------|------------------|
| 01 | chaoliu.ln.pic | 264            | 3700           | 1300             |
| 02 | mochouhucz.fac.pic | 242            | 4100           | 1400             |
| 03 | pliujun.ln.pic  | 392            | 2500           | 1350             |
| 04 | Biandian_500k.pic | 542            | 2800           | 1300             |

5. Conclusion
Faced with the growing demand for information monitoring and query through convenient mobile applications anytime and anywhere in the current grid control system, if developers still maintain the traditional "one area, one application" development framework model, it is obviously unable to meet this increasing demand. Customer needs are changed. The research on power grid regulation and control mobile application release management technology proposed in this paper adopts the design mode of "shared application + personalized interface", and can realize the effect of customers' independent editing and updating of pages without excessive intervention by developers; Cache of graphics files, refreshing of the visible area during operation, and other mechanisms to improve the opening time of the graphic screen and the response speed during operation; through the publish/subscribe message bus mechanism, the hot update effect of the interface is realized; through one-key recovery Services provide users with more operating space.

References
[1] Shi, W.M. (2016) Introduction to Android App Development. China Machine Press, Beijing.
[2] CSDN. (2018) Delve into the Java thread pool -the Executor framework. https://blog.csdn.net/weixin_33813128/article/details/89656421.
[3] Jian Shu. (2018) ASimpleCache. https://www.jianshu.com/p/3596cf09bc02.
[4] Liu, H.F. (2017) JD's road to basics architecture. Publishing House of Electronics Industry, Beijing
[5] Ou, Y.S. (2021) Android APP development introduction and project practice. Tsinghua University Press, Beijing
[6] Zhai, M.Y. (2020) Key Technology And Application For Power Transmission And Transformation Equipment. China Electric Power Press, Beijing
[7] Fang, C.S., Huang, K., He, Y. (2019) Research And Implementation of Power Grid Visualization Platform Based on GIS. Information Technology, 03: 129-133.