A web application for visualization, analysis, and processing of agricultural monitoring spatial-temporal data

Alexander A. Pushkarev[1], Oleg E. Yakubailik[2]

[1]*Federal Research Center Krasnoyarsk Science Center SB RAS, Krasnoyarsk, Russia*
[2]*Institute of Computational Modelling SB RAS, Krasnoyarsk, Russia*

Abstract
The paper discusses some features of the client and server implementation of a web application for visualization, analysis, and processing of spatial-temporal data using the react JavaScript library and the organization of a software infrastructure for convenient development using the Redux library. Architectural solutions for building the system are presented. Further development plans are described.

Keywords
JavaScript, React.js, Node.js, Web GIS, geospatial data, web mapping, thematic layer, geoportal.

1. Introduction

The development of software solutions for visualization, processing, and spatial-temporal data analysis gradually changes its vector of development from desktop programs towards web technologies and cloud computing. The growth of the computing power of personal computers and the increase in the bandwidth of the global Internet network allows you to perform complex operations directly on the client-side of web applications. Some existing systems that perform basic and complex operations on spatial-temporal data are presented in [1, 2]. This paper discusses some features of the client and server implementation of a web application for visualization and analysis of heterogeneous spatial and temporal data of agricultural monitoring using the React and OpenLayers JavaScript libraries.

Data sources for the designed system can be servers that provide spatially linked images available for visualization using various JavaScript mapping libraries (for example, OpenLayers, Leaflet, etc.). The corresponding functionality is provided in the Web Mapping Service (WMS) protocol of the OpenGeoSpatial Consortium. The system can also visualize and perform various manipulations with vector data using the previously developed online library and other libraries (for example, Turf.js, D3, etc.).

However, as the main data sources for the developed system, we can single out the ICM SB RAS geoportal [3], which provides an API for accessing catalogs and portal resources, and provides cartographic data using the WMS protocol. The API of the ICM SB RAS geoportal is

SDM-2021: All-Russian conference, August 24–27, 2021, Novosibirsk, Russia
pushkarev@ksc.krasn.ru (A. A. Pushkarev); oleg@icm.krasn.ru (O. E. Yakubailik)
0000-0002-2668-4776 (O. E. Yakubailik)

closed. It is available only to those users and developers who have an account on the geoportal. To get catalogs and resources, it is necessary to specify an active session in each request to the geoportal API.

The second main source will be the AGROCASH database system currently being created at the Krasnoyarsk Scientific Center of the SB RAS with the participation of the authors [4]. This system provides data on agricultural land obtained as a result of processing the initial data of MODIS satellite images to any territory within the studied area and for any period of time in the format of JavaScript arrays. The interface provides the ability to build parametric program queries to stored data. This functionality allows various kinds of manipulations with data on the web application client-side, for example, the calculation of various spectral indexes (NDVI, NDSI, etc.), the construction of isolines, or isosurfaces, etc.

2. Software architecture

Architecturally, the system is a web application that can connect to various external data sources that provide spatial information using specialized software interfaces (APIs). The general scheme is shown in Figure 1.

The server part of the web application is written in Node.js using the Express framework, and the client part is implemented using the React library using Redux technology to store the application state. The MongoDB NoSQL database is used as a database for storing partitions and projects created and viewed in the application.

3. Server-side of web application

The server side of the web application is written in Node.js. This allows you to use a single programming language during the entire development of a web application, which significantly speeds up and facilitates the development. This application does not require the construction of a complex architecture of data storage and access, systems of complex mathematical data

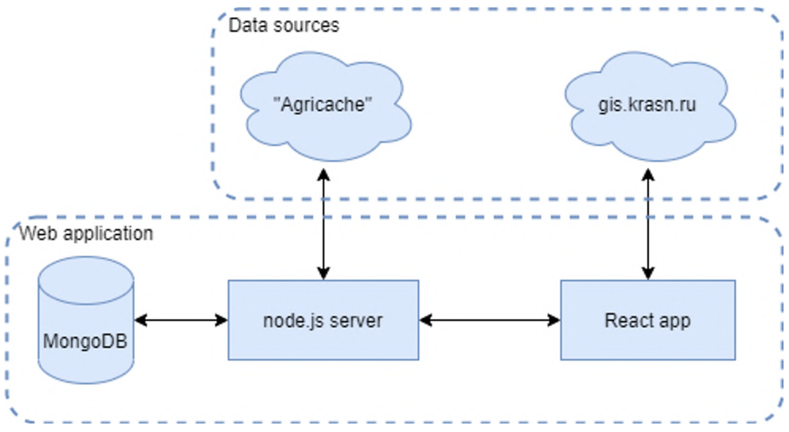

Figure 1: General diagram of system software components.

processing, and other tasks that required the use of more complex technologies for building the server part. In contrast, for example, to ASP.NET or Java, which are used to build large and complex server architectures with many complex interactions between many databases and other servers, or Python, which is well suited for data processing and machine learning.

The main function of the web application server is to process client requests, since the main business logic is focused on the client part, which allows you to save server computing resources and not use expensive server equipment for this system.

The server contains schemas of data structures stored in the MongoDB database, namely, a partition schema containing the name and description of the partitions and a project schema, which in addition to the name and description contains the initial state of the map (zoom level, coordinate center), as well as abounding area for layers, the initial state of the timeline and a list of project resources, which itself is a separate schema. The schema of a separate resource stores its name, and a unique identifier used to form a wms request to the ICM SB RAS geoportal (if it is a geoportal resource), as well as the initial parameters of the layer itself, such as visibility, transparency, and position relative to other layers. These schemas are necessary for storing partitions and projects in the MongoDB database, accessed using the Mongoose library, which provides a convenient API for interacting with MongoDB from node.js.

One of the most important modules of the server is the router. The router processes HTTP requests from the client and interacts with the database based on these requests. There are 4 types of requests: GET, POST, PUT, DELETE, each of which is used for a specific procedure (Figure 2 schematically shows the interaction of server modules with each other).

From the GET requests responsible for getting any data, the application implements request handlers for getting all the sections of the catalog, getting one section by a unique identifier, getting a list of projects located in the catalog, and getting a separate project by ID.

POST requests are responsible for transmitting data to the server, creating new entities in the database, or processing the data transmitted to the server. The application implements such requests for creating new catalog sections and creating new projects, PUT and DELETE requests are responsible for changing and deleting existing entities in the database, respectively. This application implements requests for changing and deleting catalog sections and changing and deleting projects.

The handler of each request is accompanied by validation of the data received in the request

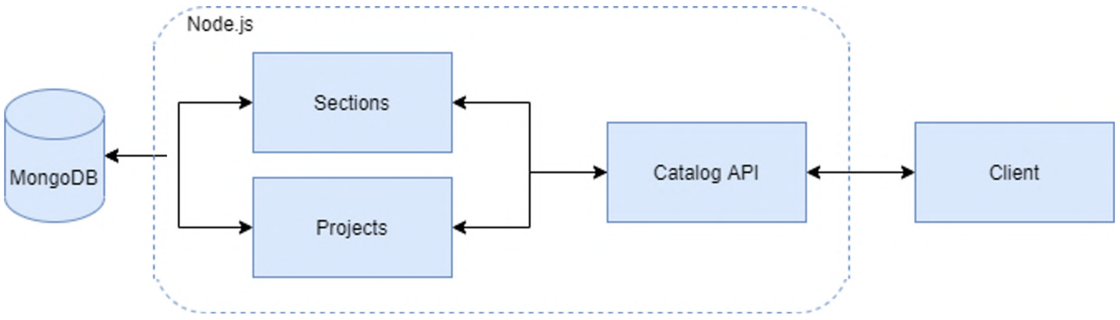

Figure 2: General diagram of web application server-side.

and verification of the user's access rights. These procedures are necessary to protect the database from getting unwanted information and restrict access to it from users with insufficient rights.

4. Client-side of web application

The client part of the web application is the face of the entire system; it is with it that the system users interact. With the development of personal computer performance and the emergence of new Internet standards, the emphasis on developing small and medium-sized projects is rapidly shifting towards single-page web applications (SPA). Such applications often perform many functions that were previously accepted to perform on the server, for example, various mathematical calculations, rendering web pages, etc.

This project uses the React library for fast and high-quality development. This library is developed by Facebook and is the most popular solution for building single-page client web applications. Due to its flexibility (React is a library and does not dictate strict development rules, such as frameworks such as Angular and Vue), React allows you to quickly develop small and medium-sized projects that will be quite easy to maintain and expand in the future.

The client application includes several pages with reactive switching (without reloading the page). This article will discuss in detail the pages of the catalog and the map editor. A simplified diagram of the React application is shown in Figure 3.

The catalog page contains section cards that contain the names and descriptions of these sections. Users with sufficient rights can create and modify sections using special dialog boxes. When you go to a separate section, project cards are displayed, containing the name and description. Users with sufficient rights can create new projects, and ordinary users and guests can only view existing projects without the possibility of changing them.

On the project page, there is a map with various tools for processing layers and a sidebar on which there are map layers and tools for enabling/disabling these layers, changing visibility, and arranging the display order of all layers.

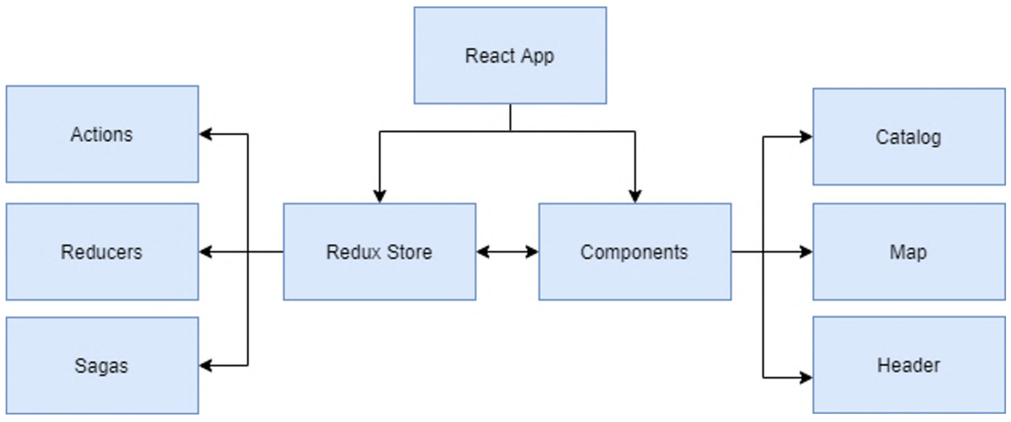

Figure 3: General diagram of web application client-side.

When editing or creating a project, you can add new layers to the map. When you click on the corresponding button, a pop-up window opens in which you need to select a data source (at the moment, only the ICM SB RAS geoportal is available). When switching to the ICM SB RAS geoportal data source, the directory tree of the geoportal available to the current user is displayed. When you select a specific resource from the catalog, you can view its metadata and change the layer's name in the system. After adding a layer, it is displayed in the side menu, where you can perform various manipulations to change the parameters of its visualization.

All layers located on the map are stored as in the application's state, for which the Redux library is responsible. This library allows you to conveniently manipulate local application data from any React component, which greatly simplifies development and increases code readability and further scaling.

Redux is designed to have storage and a set of functions (actions) that allow you to interact with the storage. For example, for a project page, actions such as adding a layer, updating the visibility, transparency of the layer, updating the order of all layers, changing the visibility boundaries of layers, etc., are implemented. Each of these actions changes the array of layers located in the Redux storage, based on which the layers are displayed on the map using the OpenLayers library.

OpenLayers with the ol-ext extension package is used as a mapping library. This library is the most powerful mapping tool with many features, but there are other JavaScript mapping libraries, such as Leaflet. This library is much more convenient to use than OpenLayers since it provides an official library for React. Still, it has some disadvantages when working with a large number of detailed vector layers, making it unsuitable for the tasks that the system being developed is aimed at. However, Leaflet is great for simple geospatial web applications.

A separate component has been developed to work with OpenLayers, a kind of wrapper over OpenLayers since the map object must be saved and not redrawn every time the component is rendered. A set of hook components is implemented for individual OpenLayers entities, such as layers, events, and tools, is implemented. All these components are collected in a mini library and can be reused in other projects.

The system can view data that changes over time (provided that such data is available). The time-bound data will automatically change depending on the selected time interval by moving the slider on the timeline.

It is also worth noting that all layers and map parameters are saved in the HTTP query parameters, allowing you to save the project's current state when the page is reloaded. This feature allows you to save the state of the project in case of closing the browser or accidentally switching to another page and also allows you to exchange user settings of the project by sending just a link to the project.

5. Results

The functionality of the web application developed now allows you to create cartographic projects based on the ICM SB RAS geoportal data, with the possibility of interactive interaction with map layers (enabling/disabling the layer, changing the transparency, the order of layers, etc.). A system for cataloging projects by sections is also implemented, with the ability to create,

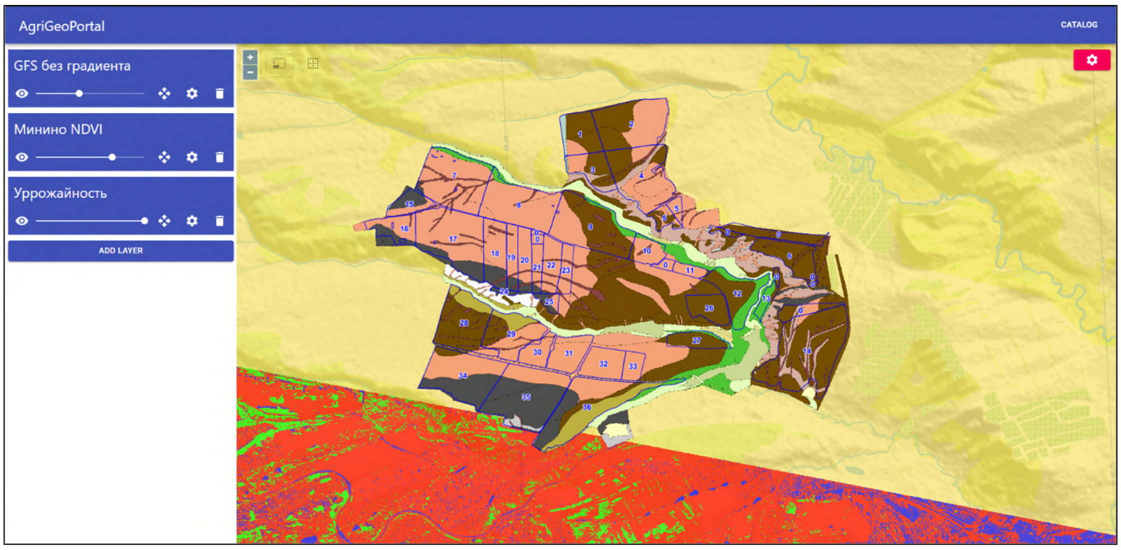

Figure 4: Project creation/viewing page interface.

delete and modify sections and projects. In the future, it is planned to add support for other data sources for the system, to create more detailed and complex cartographic projects.

Figure 4 shows the project creation/viewing page interface, to which 3 layers have been added, and certain display parameters for each layer are set separately. The display area is also selected, which restricts the display of all layers to a rectangular area.

At this stage, the system does not have the ability to perform complex operations on map layers. Still, in the future, after other data sources are added to the system that will provide vector information, the system will have tools that allow you to calculate various interpolations from point data, perform mathematical calculations on grid data, as well as perform statistical operations on data and visualize the results on graphs.

Also, the system will have an authorization subsystem that will use a microservice architecture. This authorization subsystem will be synchronized with the ICM SB RAS geoportal, allowing users to access this geoportal's resources.

6. Conclusions

The paper considers some features of the architecture of a cartographic web application for visualization, analysis, and processing of spatio-temporal data. Functional components of the client application are implemented for the collaboration of several JavaScript libraries.

The web application's server-side has also been developed to process requests from the client-side of the web application and store a catalog of resources and projects in the MongoDB database. The diagrams reflecting the general architecture of each side of the web application are presented, and the interest of the application is also demonstrated.

References

[1] H. Tamiminia, B. Salehi, M. Mahdianpari, L. Quackenbush, S. Adeli, B. Brisco, Google Earth Engine for geo-big data applications: A meta-analysis and systematic review, ISPRS Journal of Photogrammetry and Remote Sensing 164 (2020) 152–170. doi:10.1016/j.isprsjprs.2020.04.001.

[2] L. Zhu, X. Chen, Z. Li, Multiple-view geospatial comparison using web-based virtual globes, ISPRS Journal of Photogrammetry and Remote Sensing 156 (2019) 235–246. doi:10.1016/j.isprsjprs.2019.08.016.

[3] O. Yakubailik, A. Kadochnikov, A. Tokarev, Applied software tools and services for rapid web GIS development, volume 1. No. 2, 2015.

[4] O. Yakubailik, A. Kadochnikov, A. Tokarev, The system of operational processing of satellite remote sensing data in Krasnoyarsk science center of Russian Academy of Sciences, in: 5th International Conference Information Technologies in Earth Sciences and Applications for Geology, Mining and Economy, ITES and MP 2019, Moscow, CEUR Workshop Proceedings, volume 2527, 2019, pp. 47–51.