UDC 004.418

Kudrenko S.A.,
orcid.org/0000-0002-0759-3908,
Zhuravel C.V.,
orcid.org/0000-0001-6536-6298,
Fomina N.B.,
orcid.org/0000-0003-2357-6371

OVERVIEW AND JUSTIFICATION FOR CHOOSING TECHNOLOGY STACK FOR DATA ANALYSIS SYSTEM

National Aviation University

stanislava@i.ua
sergey.zhuravel@gmail.com
nfomina@ukr.net

Introduction

All management tasks require accurate analysis. And, since earlier dozens of factors that a person could calculate manually took part in the analysis of the situation, now there are much more of these factors. In connection with all this, there is a need to automate the collection and processing of sociological information. The main advantage of the automation process is that it allows to reduce the amount of required memory, reduce the time for data processing, and reduce the number of copies of documents when updating information. Also, as for advantages, it allows to find the information of interest in a short period of time.

The introduction of the automation process will simplify the researcher's work and will allow him to perform his work much faster, more complete and better.

Also, one of the requirements for the system being developed is the creation of a convenient user interface that is not overloaded with unnecessary information and functionality.

It provides easy perception and processing of information, providing a free output of information in a more familiar form for employees of the enterprise, such as in document format. The common assignment of the task is to design and develop a project, that will be suitable and comfortable to work both with the small amount of data required for concrete issue and with a huge amount of data.

Problem statement

The common assignment of the paper is preliminary analysis of requirements and technologies for design and develop a project, that will be suitable and comfortable to work both with the small amount of data required for concrete issue and with a huge amount of data.

The choice of technologies for developing an application is an important stage which has been described in paper. Before developing the analyze data system, the requirements should be carefully prepared and described. A well-chosen combination of technologies should ensure comfortable work in the future at all stages of the application's existence

System Requirements

Performance Requirements. Performance requirements determine the effectiveness of the function’s execution under the pointed set of conditions. There are a lot of acceptance points, which should be reviewed on the performance issues. For example, load testing is used for checking the speed of how long the application runs under the expected user loads. The aim of load testing is to find dangerous parts before the application deployed.

The following list of performance characteristics should be taken care of while developing the system:

- user-friendliness: the system functionality and design should be intuitive understandable so new users should not have any problems with onboarding;
- user satisfaction: the system should match the user expectations. If the performance results do not coincide with the requirements, then it should be optimized;
- response time: the system should react to the user's actions as quickly as possible. Ideally, the callback should be executed immediately. For some asynchronous actions that cannot provide feedback in a nutshell, a load UI element must be added during action processing.

*Functional Requirements***.** Functional requirements define the functionality of the software. They are sometimes called behavioral requirements. The following requirement are defined for the current system. The system should have the platform menu in the header and check whether users have access to surveys and dashboard.

System will have a functionality to download survey question dashboards and charts with data in .pptx, .jpg, .jpeg, .png formats.

There should be a possibility to export all questions as a .pptx slide deck in a standardized format.

Question block should provide the general information related to a question and a statistical chart that displays survey data. User will have the possibility to sort data bars via drag and drop, change chart axes and keys, add average lines, export chat to .png., add confidence intervals for bars and values and show data as percentages.

*Interface Requirements***.** When developing a user interface, it necessary to be guided by the UX approach and the tasks that the interface should solve.

UX is a term that describes the degree of user satisfaction from using the product.

Tasks which should be solved by the user interface:

- user authorization;
- granting user access to the resource and personal account;
- possibility to be tested;
- displaying the result.

As a result, the main sections of the application were highlighted:

- dashboard page;
- login page;
- personal account with information about the user;
- survey page;
- question analysis page.

For the direct development, there should be used component approach.

The component approach is a programming paradigm that essentially relies on the concept of a component - an independent module of program code intended for construction. For these purposes, the React.js library is ideal because it supports the component approach.

There should be uniform, synchronized data between the sections and in the application as a whole. For this purpose, the schema of several design patterns will be used. Application model, user interface and user interaction are divided into three separate components. So that modification of one of the components has minimal impact on the others. This schema forms a MVC binding.

The main purpose of applying this concept is to separate the business logic from its visualization.

Security Requirements. Besides fact that the system should be convenient to use, it also has to provide security to users and their private data. In order to supply the system safety and reliability the developer should:

- add logic to validate the input value;
- provide blocking wrong user actions which are out of the scope;
- ensure the integrity of stored data;
- exclude the possibility of the unauthorized access to the information.

The peculiarity of hacking resources that it is automated, but not personalized. It carried out in large quantities using special programs. The owner of resource which works with clients on the model of a web application must be able to protect the project from the most common methods of hacking.

Before developing a methodology for protecting a web application from potential

threats, the site should be checked for vulnerabilities. The check is carried out manually or automatically.

The programs will test the application for major risks. These software products exist in two versions: black hat, simulating the actions of hackers, and white hat, systematically revealing all system flaws by scanning.

Some of the most effective tools are:

- scan local networks for vulnerabilities;
- looking for the possibility of injecting malicious code into a web page that steals user account data and other information. The code is injected through vulnerabilities on the user's server or device;
- examine the configuration of a web application and find redundant or malicious code;

Working with web services requires the use of a wide range of security tools. In addition to the main listed methods, the following are often used:

- passwords encryption;
- avoiding cross-site scripting;
- control of uploading files to the server.

Technologies and Platforms Selected for Creating the System

The choice of a technology stack must be approached very carefully and responsively. It is necessary to look far ahead into the future and predict the potential development and fate of the project. Obviously, the stack should be easily scalable, functional, correspond the latest market trends. It should meet the most modern features. Most importantly, it has to be easily supported in the future by other developers. The presence of a large community of developers in the world for this or that product in the stack and open source code are huge advantages that should be cared. Also, these technologies should not contradict each other. Their existence together should be harmonious and justified. So, for example, GraphQL is great for generating queries in JavaScript. It also uses a data schema that is easy to use in JavaScript.

Guided by the above factors, as well as listening to the opinions of leading developers of the web technology market, there were identified the following technology stack:

- React.js library;
- GraphQL library;
- Webpack - the utility is great for modular building of a web application;
- npm package manager for managing modules and dependencies.

React.js Library

React.js has a capacious and understandable Application Programming Interface (API). To work with React, it is necessary to understand a number of terms and the differences between them. Elements are JavaScript objects that are HTML elements. Components are elements of React.js that are created by the developer and can have any name. As a rule, they contain their own specific structure and perform a number of functions. React elements and components are built using JSX. JSX is a deep JavaScript syntax that looks like XML.[1]

React creates an analogue of the real Document Object Model (DOM) tree from components - VirtualDOM and presents it in the browser. The library monitors changes in the virtual tree and, when it changes, updates the real DOM so that the real and virtual tree are the same. (fig. 1)

Above are the basic concepts it is needed to know to get started with React.js. Also, there is good to mention about the existence of state in React.

In practice, in the project it will be implementing a state of the application using GraphQL. React and GraphQL is a fairly successful alliance that occurs quite often.

Components have a life cycle such as mount, update and unmount. The library provides the ability to define various points in the life cycles of components and interact with them. When the first use of the component, call the lifecycle methods in this order:

- constructor;
- getDerivedStateFromProps;
- render;
- componentDidMount.

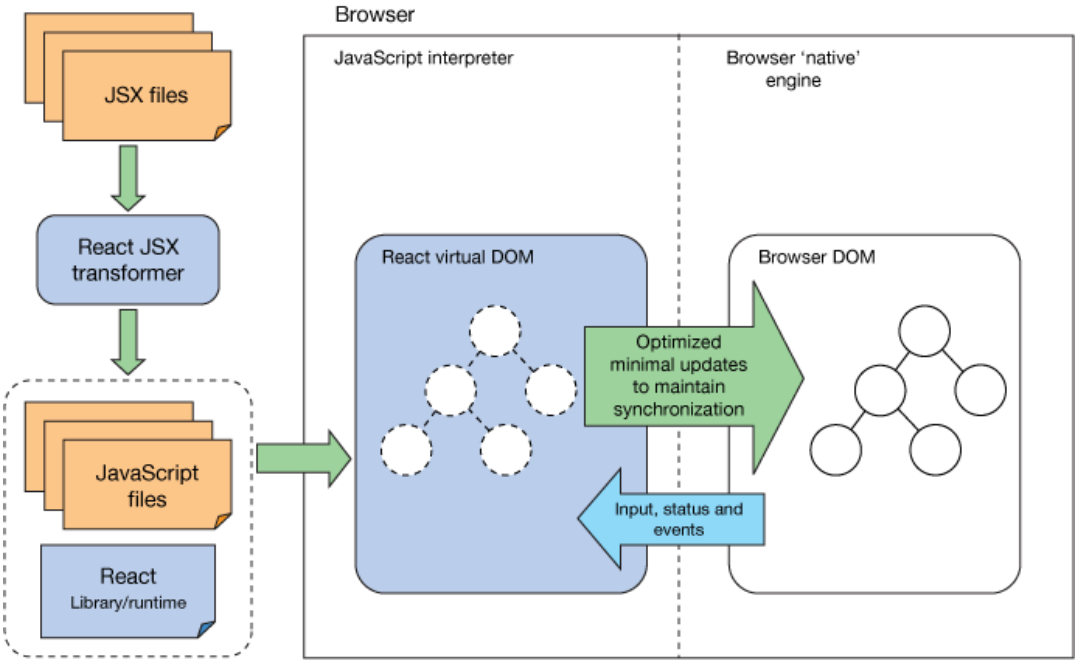

Fig. 1. React operations

Why React.js Library was integrated:

- The content is referenceable

This is the functionality that makes the difference compared to other

frameworks. Thanks to the use of a Node server, the code will be able to be generated on the client side and on the server side. Unlike other traditional JS frameworks which natively execute code only on the client side in the browser.

- React.js is very fast

React.js creates its own virtual DOM where the components are attached. This approach gives a tremendous amount of flexibility and exceptional performance, as React.js calculates which changes in the DOM needs to be made, and just changes the part that needs updating. In this way, React.js avoids expensive operations in the DOM.

- Components are the future of web development

React.js took the concept of Shadow DOM and the Polymer.js framework and took it to the next level.

React.js doesn't use Shadow DOM instead it gives the ability to create the components that can later be reused, combined, and included in the core content. This functionality alone is a guarantee of productivity by the ease of defining and manipulating developer's own components.

- Intelligibility

React.js produces easy to read code, reading it immediately determines what the

functionality of the application is. That is essential for the maintenance and expansion of the project over time.

- Javascript is easier to write

ReactJS uses a special syntax called JSX, which allows to mix HTML and JavaScript. This is not required it can still be written the React.js app in native JavaScript but this new syntax allows to write the components very easily. Being able to put a touch of HTML in the rendering functions. And after a while it becomes very natural.

On the image below there is an official logo of React.js library.

Typescript Programming Language. The TypeScript language is one of the most popular technologies of recent years, both in Frontend and Backend development. Its popularity continues to grow and it is at the heart of many projects: Angular, NativeScript, Ionic, VS Code, Apollo GraphQL, Babylon.js, RxJS, Nest, TypeORM, etc.

TypeScript is an opensource language, created by Microsoft, stackable in JavaScript.

On the image below there is the logo of Typescript.js library.

Its main objectives are:

- support for existing and future EcmaScript proposals;
- the contribution of optional typing to JavaScript;
- the early identification of potentially invalid codes;
- compilation to optimized JavaScript, with a target choice: ES3, ES5, ES6 and next ones.

What TypeScript does not aspire to do, among others:

- imitate existing languages; but rather exploit the nature of JavaScript and the uses of developers as a guide to make the language relevant;
- use a sage type system [3].

TypeScript's type system is not wise, which means that it allows certain operations that cannot be verified at compile time. This is one of the big major differences with Flow that can be considered.

GraphQL Query Language. GraphQL is a query and data manipulation language for APIs, as well as an environment for making those queries. The language was developed in 2012 by Facebook for the internal needs of the company. [4-5]

Today, GraphQL is used in many popular applications. First, the social network Facebook. The GraphQL is used in products such as Airbnb, GitHub, Pinterest, Shopify, New York Times and many others.

The fact that it was created specifically on Facebook for a project with a large amount of heterogeneous data ensure that working on a product it should not arise with the limitations of the REST architecture.

For example, getting a user's profile, posts and comments does not initially seem difficult. But if it is considered the amount of data in the system and assumed that all this data is stored in different databases (for example, MySQL and MongoDB), it becomes clear that this will require creating several REST endpoints.

Imaginings how large the volume of data and heterogeneous data sources are, it becomes clear why it was necessary to develop a new approach to working with the API. This approach is based on the following principle: it is better to have one smart endpoint that will be able to work with complex queries and return data in the exact form and volume that the client needs.

At the heart of any implementation of the GraphQL API is the data schema - this is a description of what data types it can work with and what data types it can return in response to a request - they are described in the GraphQL type system. [4]

To work with any API, the developer needs to know what types of objects can be obtained, what fields to select, what fields are available in internal objects and more. The GraphQL schema is showed on figure 2. Working with the GraphQL API, the client does not care at all where the data comes from.

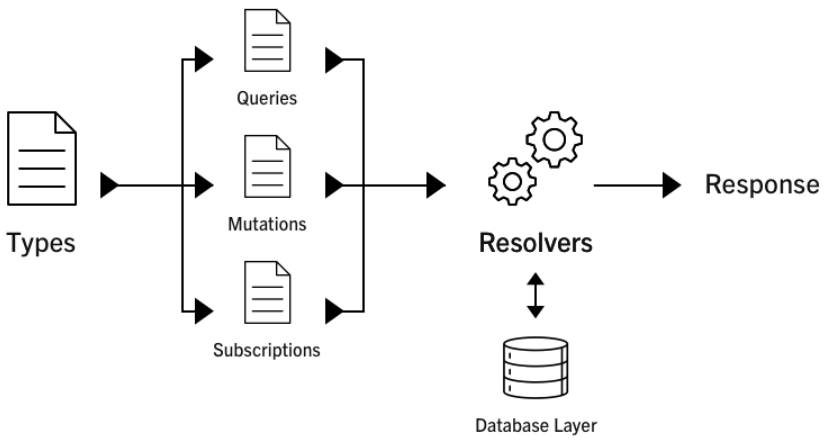

Fig. 2. GraphQL schema

It just should be made the request to the extent is needed, and the GraphQL server returns the result. Therefore, it can be concluded that the schema is a contract between the API and the client. Before the client makes any request, it is validated in accordance with the schema of the given API.

Many platforms now support GraphQL: web, Android, iOS and others. The GraphQL client sends a request to receive data or to change it, composed in accordance with the schema, to the GraphQL server.

The GraphQL server is the HTTP server to which the GraphQL schema is associated. It means that all requests received from the client and returned responses are passed through this scheme.

The GraphQL server cannot know what to do with a request unless it is explained to it using special functions. Due to them, GraphQL understands how to get data for the requested fields.

These functions are associated with the corresponding fields and are called resolvers. A response is returned to the client that reflects the data structure requested from the client, usually in JSON format.

It is possible to work with completely different data sources: databases (relational / NoSQL), web search results, Docker, and other. On the image below there is an official logo of GraphQL library.

D3.js Library. D3.js is a JavaScript library for data processing and visualization. The name D3 itself stands for Data-Driven Documents and, as it were, focuses on data management, although the key functionality of the library is its powerful visualization capabilities [6].

The D3.js library is primarily based on JavaScript, SVG and CSS, as opposed to other similar libraries that use the canvas element and its capabilities instead of SVG.

While standard rendering engines like the canvas element rely on pixels, svg uses vectors. Using SVG allows to create graphically rich structures with animation and interoperability. Compared to pixel art, SVG has several advantages.

In particular, SVG is based on xml, which makes it more readable. In addition, SVG code is more lightweight than image files. Largely due to this, D3 is currently one of the most popular frameworks used for graphical data processing and creating all kinds of charts and graphs. On the image below there is an official logo of D3.js library. Unlike other similar JavaScript libraries, D3 does not use jQuery to work with the DOM structure, although at the same time it implements similar concepts for working with elements.

React Testing Library. One of the convenient solutions for unit testing of components is the react-testing-library built on top of react-dom and react-dom/test-utils. It provides utility functions on top of react-dom. Tests run on DOM nodes, not React component instances. The main idea of the library is to bring test scripts closer to using the components the way the user does, which allows for confidence, giving more confidence when the application hits production. It is a lightweight solution for testing React components.

The Jest test framework will be used to run the tests. It is built in such a way that it requires almost no configuration, it will not be difficult to install it in a new project and start using it, moreover, it is part of create-react-app. It is also needed helper library that provides custom DOM mapping for Jest. On the image below there is an official logo of React Testing Library and Jest.

The react-testing-library contains all the necessary methods for testing components, simplifies testing and improves test readability.

Conclusions

It is necessary to look far ahead into the future and predict the potential development and fate of the project. Obviously, the stack should be easily scalable, functional, correspond the latest market trends. It should meet the most modern features. Most importantly, it has to be easily supported in the future by other developers.

React.js has a capacious and understandable API. To work with React, it is necessary to understand a number of terms and

the differences between them. Elements are JavaScript objects that are HTML elements. Components are elements of React.js that are created by the developer and can have any name. As a rule, they contain their own specific structure and perform a number of functions.

The TypeScript language is one of the most popular technologies of recent years, both in Frontend and Backend development. Its popularity continues to grow and it is at the heart of many projects.

GraphQL is a query and data manipulation language for APIs, as well as an environment for making those queries. At the heart of any implementation of the GraphQL API is the data schema - this is a description of what data types it can work with and what data types it can return in response to a request. When working with the GraphQL API, the client does not care at all where the data he requests comes from. It just makes the request to the extent it needs, and the GraphQL server returns the result.

D3.js is a JavaScript library for data processing and visualization. The name D3 itself stands for data driven documents and focuses on data management, although the key functionality of the library is its powerful visualization capabilities.

References

1. *Johnson, Nicholas.*"Introduction to Flux – React Exercise" [Electronic resource]. Access mode: http://nicholasjohnson.com/react/course/exercises/flux/.

2. A JavaScript library for building user interfaces [Electronic resource]. Access mode: https://reactjs.org/.

3. *Anders Hejlsberg.* What is TypeScript and why with Anders Hejlsberg [Electronic resource]. Access mode: www.hanselminutes.com.

4. [Electronic resource]. Access mode: https://graphql.org/.

5. "Why use GraphQL, good and bad reasons". *Honest Engineering*. 4 August 2018. [Electronic resource]. Access mode: https://honest.engineering/posts/why-use-graphql-good-and-bad-reasons

6. *Kudrenko S.A.* Method for complex objects automated design on autodesk revit based // Kudrenko S.A., Fomina N.B., Kramarenko I.P. // Проблеми інформатизації та управління. – №63. – P. 64-74.

Kudrenko S.A., Zhuravel C.V., Fomina N.B.

OVERVIEW AND JUSTIFICATION FOR CHOOSING TECHNOLOGY STACK FOR DATA ANALYSIS SYSTEM

The main advantage of the automation process is that it allows to reduce the amount of required memory, reduce the time for data processing, and reduce the number of copies of documents when updating information.

The choice of technologies for developing an application is an important stage which has been described in paper. Before developing the analyze data system, the requirements should be carefully prepared and described. A well-chosen combination of technologies should ensure comfortable work in the future at all stages of the application's existence

Obviously, the technology stack should be easily scalable, functional, correspond the latest market trends. It should meet the most modern features. Most importantly, it has to be easily supported in the future by other developers.

React.js has a capacious and understandable API. To work with React, it is necessary to understand a number of terms and the differences between them. The TypeScript language is one of the most popular technologies of recent years, both in Frontend and Backend development. Its popularity continues to grow and it is at the heart of many projects. GraphQL is a query and data manipulation language for APIs. The name D3 itself stands for data driven documents and focuses on data management.

Keywords: *automation data analysis, Application Programming Interface, GraphQL, React.js library.*

Кудренко С.О., Журавель С.В., Фоміна Н.Б.

ОГЛЯД І ОБҐРУНТУВАННЯ ВИБОРУ СТЕКУ ТЕХНОЛОГІЙ ДЛЯ СИСТЕМИ АНАЛІЗУ ДАНИХ

Головною перевагою процесу автоматизації є те, що він дозволяє зменшити обсяг необхідної пам'яті, скоротити час на обробку даних та зменшити кількість копій документів при оновленні інформації.

Вибір технологій для розробки додатків є важливим етапом, який був описаний у роботі. Перш ніж розробляти систему аналізу даних, слід ретельно підготувати та описати вимоги. Правильно підібрана комбінація технологій повинна забезпечити комфортну роботу в майбутньому на всіх етапах існування програми

Очевидно, що стек технологій повинен бути легко масштабованим, функціональним, відповідати останнім тенденціям ринку. Він повинен відповідати найсучаснішим характеристикам. Найголовніше, що в майбутньому його повинні легко підтримувати інші розробники.

React.js має місткий і зрозумілий API. Для роботи з React необхідно розуміти ряд термінів та відмінності між ними. Мова TypeScript - одна з найпопулярніших технологій останніх років, як у розробці Frontend, так і в програмі Backend. Його популярність продовжує зростати, і це в основі багатьох проектів. GraphQL - це мова запитів та обробки даних для API. Сама назва D3 означає документи, керовані даними, і зосереджена на управлінні даними.

Ключові слова*: автоматизація аналізу даних, прикладний програмний інтерфейс, GraphQL, React.js library.*