http://newsep.com.ua/new/1016.
[2] Воинова С.А. О проблеме управления эффективностью функционирования изношенного оборудования / Енергетика та електрифікація, 2016, №. 8.- С. 36 – 39.
[3] Воинова С. А. Обновление как инструмент развития производства/ Известия вузов и энергетических объединений СНГ, № 2, 2013.- С. 69 - 74.
[4] Воинова С. А. Часткове оновлення – інноваційний інструмент управління ефективністю функціонування устаткування, що відробило ресурс / Автоматизація технологічних і бізнес-процесів, 2016, Volume 8, Issue 1.- С. 71 - 76.
[5] Воинова С. А. Актуальные задачи управления экологической эффективности технических объектов/ Матер. Междун. конф. “Стратегия качества в промышленности и образования” (1-8 июня 2007г., Варна, Болгария). Дніпропетровськ - Варна: “Фортуна”. – ТУ Варна – 2007г., - Т.1. - С.102 - 104.
[6] Воинов А.П., Димитрова Ж.В., Воинова С.А. Актуальность обновления оборудования в системах централизованного теплообеспечения возрастает с ускорением / Зб. тез допов. Міжнар. наук.-практ. конф. “Сучасне місто — проблеми та їх вирішення”, 21 — 23 вересня 2017р., Одеса.- Одеса: ОДАБА.- С. 91 - 92.

UDC 519.85:004.42

OVERVIEW OF POPULAR APPROACHES IN CREATING CLIENT-SERVER APPLICATIONS BASED ON SCIENTOMETRICS ONAFTS’ PLATFORM

D. Salskyi[1], A. Kozhukhar[2], O. Olshevska[3], N. Povarova[4]

[1,2,3,4]Odessa National Academy of Food Technologies, Odessa, Ukraine
ORCID: [3]0000-0002-4512-3915, [4]0000-0003-3630-8384
Scopus ID: [3]57192687506, [4]56578764800
E-mail: [1]salsky.d@gmail.com, [2]alex.kozhuchar@gmail.com, [3]olshevska.olga@gmail.com, [4]povarova.natasha@gmail.

DOI: 10.15673/atbp.v10i4.833

Abstract: Most of the currently developed systems are based on the client-server architecture. This architecture is used everywhere, from mobile-native development to Web applications.

However implementing an application based on this architectural solution requires quite a lot of effort from the software developer, and therefore, in order to simplify and speed up the development, certain standard solutions and approaches appeared. This article will discuss the most popular technologies used in the development of Web applications in the context of enterprise development.

Also in this article will be mentioned the project, built on the architecture of "client-server" - ScienceToMetrics.

The main theme of this project is the study of science-metric indicators for the structural divisions of the faculty of the Odessa National Academy of Food Technologies. In fact, it is a portal for viewing and editing information on employees, in the future this portal may be extended to subprojects.

In this project, the main idea of this architecture was embodied: decomposition of the application into atomic parts in order to distribute it to several hardware units of capacity to improve performance. The client is an independent application, which at the same time receives information from an external API-interface through REST-requests. In turn, the backend provides this API with certain security restrictions on the content provided. The backend for this architecture provides a layer for the content of the data users, whether it's a database (NoSQL, SQL) or an integration API with external aggregation systems. To ensure the necessary level of security, JWT (Javascript Web Token) authorization is used, which allows you not to create an explicit session between the client and the backend, but allows you to communicate through a token that stores all the necessary meta-information for this user.

Аннотация: Большинство существующих в настоящее время систем основаны на архитектуре клиент-сервер. Эта архитектура используется повсеместно, от мобильно-нативной разработки до разработки веб-приложений.

Однако внедрение приложения на основе этого архитектурного решения требует от разработчика программного обеспечения больших усилий, поэтому для упрощения и ускорения разработки появились определенные стандартные решения и подходы. В этой статье будут рассмотрены самые популярные из них, используемые при разработке веб-приложений в контексте энтерпрайз разработки.

Также в этой статье будет упомянут проект, построенный на архитектуре "клиент-сервер" - ScienceToMetrics.

Основная тема данного проекта - изучения наукометрических показателей по структурным подразделениям профессорско-преподавательского состава Одесской Национальной Академии Пищевых Технологий. По сути является порталом для просмотра и редактирования информации по сотрудникам, в будущем возможны расширения данного портала под дочерние подпроекты.

В данном проекте была воплощена основная идея данной архитектуры - декомпозиция приложения на атомарные части в целях распределения на несколько аппаратных единиц мощностей для повышения производительности. Клиент является независимым приложением, который при этом получает информацию с внешнего API-интерфейса посредством REST-запросов. В свою очередь бекэнд предоставляет данное API с определенными ограничениями безопасности по предоставленному содержимому. Бэкенд в случае данной архитектуры предоставляет прослойку для содержимого данных пользователей, будь-то база данных (NoSQL,SQL) или интеграционное API с внешними системами агрегации. Для обеспечения необходимого уровня безопасности используется JWT(Javascript Web Token) авторизация, которая позволяет не создавать явную сессию между клиентом и бэкэндом, а позволяет общаться посредством токена, который хранит в себе всю необходимую мета-информацию по данному пользователю.

Keywords: dependency injection, ORM, client-server, html, css, javascript, MVC.

Ключевые слова: внедрение зависимостей, ORM, клиент-сервер, html, css, javascript, MVC.

1. Introduction

Nowadays the architecture of client-server applications has become a very popular solution among the majority of developer. This state of affairs is dictated not only by the convenience of such a solution, but also by the requirements of modern users to the systems they use. Client-server system is characterized by the presence of two interacting stand-alone processes - the client and the server, which, in general, can be performed on different computers, exchanging data over the network. Processes that implement a service, such as a file system or database service, are called servers. Processes that request services from servers by sending a request and then waiting for a response from the server are called clients.

What are the qualities that the client-server brings to the information system?

Reliability

The database server modifies the data based on the transaction engine, which gives any set of operations declared as a transaction the following properties:

- Atomicity - under any circumstances, there will be performed either all the operations or none of them; data integrity is guaranteed when the transaction is completed;
- Independence - transactions initiated by different users do not interfere in each other's affairs;
- Durability - after the transaction is completed, its results will not be lost.

The transaction mechanism supported by the database server is much more efficient than the similar mechanism in desktop DBMS, because server centrally controls the operation of transactions. In addition, in a file-server system a failure on any of the workstations can lead to loss of data and their inaccessibility to other workstations, while in the client-server system a failure on the client practically never affects the integrity of the data and their availability to other customers.

Scalability

Scalability - the ability of the system to adapt to the growing number of users and the size of the database with adequate performance improvement of the hardware platform, without replacing the software.

It is well known that the capabilities of desktop DBMS are seriously limited - it's five to seven users and 30-50 MB, respectively. This values can deviate in specific cases. Most importantly, these limits can not be overcome by increasing the capabilities of the equipment.

Systems based on database servers can support thousands of users and thousands of GB of information.

Security

The database server provides a powerful means of protecting data from unauthorized access, which is impossible in desktop DBMSs. At the same time, access rights could by administered very flexibly - up to the level of the fields of tables. In addition, you can generally prohibit direct access to the tables, interacting with the data through intermediate objects - views and stored procedures.

Flexibility

In an application that works with data, you can select three logical layers: user interface, business logic layer and data management.

The most authoritative and complete international databases are aimed at studying the scientific activity of countries (scientists, organizations) according to bibliometric indicators: Web of Science (WoS), Philadelphia Institute of Scientific Information of Thomson Reuters Corporation and Scopus, publishing company Elsevier. Their data is taken into account in various international and national rating systems [9].

Among the noncommercial science-based databases on technical sciences, the following bases can be named: Index Coordinates, BASE, DOAJ, Driver, FreeFullPDF, UlrichsWeb and others.

2. Formulation of the problem

The main goal of this work is to develop a software product that will provide support for the preservation of science metrics

data of the faculty.

To achieve the goal following tasks were highlighted: to analyze the main problems of the subject area; analysis of existing analogues; the analysis and justification of the selection of means of implementation is carried out; the software product was developed, which provides support for the preservation of scientific and experimental data of the faculty.

3. Materials and Methods

In the analysis process of the main problems of the subject area, software analogues and means of development a systematic approach was used, object-oriented approach was used to construct the information model of the system, to create database storage NoSQL technologies was prefered. Developing process was realized with the help of the following tools: MongoDB as a free and open-source cross-platform document-oriented database, classified as NoSQL type, with the usage of JSON-like documents representation; Intellij Idea as a Java IDE; Intellij WebStorm as a JavaScript front-end IDE. The usage of Spring framework, allowed to pay more attention on business-logic of the application.

4. The main part

4.1. Server side best practices

The first thing to talk about will be one of the most popular approaches for developing Java web applications known as Dependency Injection (hereinafter DI).

Inversion of Control is an important principle of object-oriented programming, used to reduce dependency between components in computer programs and is one of the five most important principles of SOLID. The most popular implementation of IoC is the Dependency Injection. Dependency implementation is used in many frameworks, which are called IoC-containers.

The main tasks of DI are reduce dependency level and simplify testing.

The upper layer modules should not depend on the modules of the lower layers. Both types of modules should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

If the object x (class X) calls the methods of the object y (class Y), then X depends on Y. The dependency can be reversed by the introduction of the third class, namely the interface class I, which must contain all the methods that x can call for the object y. In addition, Y must implement the interface I. X and Y now both depend on I, and the class X is no longer dependent on the class Y; It is assumed that X does not realize I.

The elimination of the dependence of the class X on Y by the introduction of the interface I is called Inversion of Control (or Dependency Injection (DI)).

Let's consider an example: suppose we have a class A that explicitly has dependencies on ServiceA and ServiceB (Fig.1).

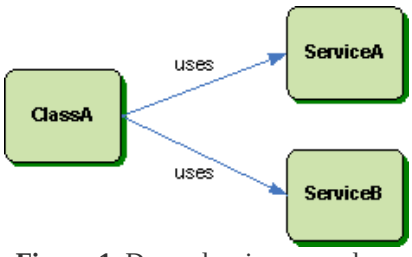

Figure 1. Dependencies example.

This brings the following problems:

- If you need to change the dependency, you need to recompile the source code.
- It is difficult to test classes in an isolation environment, since explicit dependencies can not be replaced by mocks and stubs.
- Classes contain a repeating code of creation, defining, and managing their dependencies.
- Usage of factories (Abstract Factory & Factory Method), forms a tight binding due to which, after making changes to the dependencies, it is necessary to recompile the program.

The most well-known IoC implementations (perhaps even the only ones) are Dependency Injection and Service Locator (Fig.2).

Dependency Injection plays the role of implementer of the dependencies as a separate component, which, depending on the metadata it has been provided, itself decides which dependency to implement into the class object. It is possible to develop such a component on your own, but mostly developers prefer to use components developed by third-party developers.

The most popular DI implementations at the moment are Spring Framework and Google Guice.

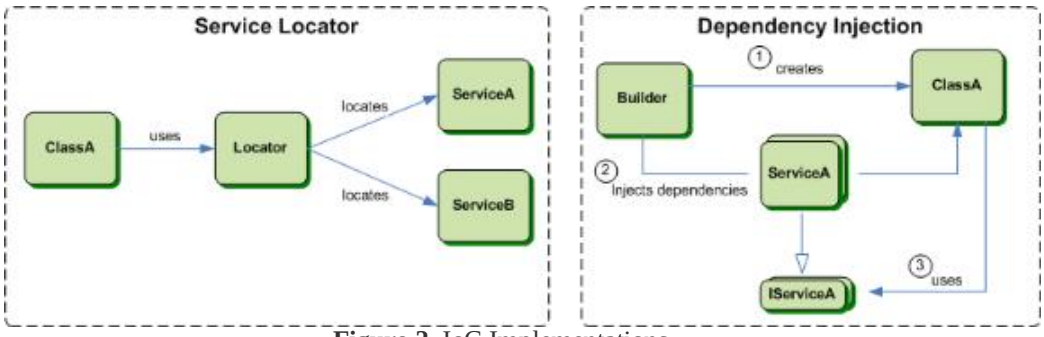

Figure 2. IoC Implementations.

Also recently, the usage of ORM frameworks in the developing web applications has become very popular.

ORM is a programming technology that links databases to the concepts of object-oriented programming languages, creating a "virtual object database". It allows you to convert data from a relational representation to an object-oriented one and vice versa. ORM is used to simplify the process of storing objects in a relational database and retrieving them, ORM itself takes care of converting data between two incompatible states. Most ORM tools rely heavily on database and object metadata, so objects do not need to know anything about the structure of the database, and the database does nothing about how the data is organized in the application. ORM provides a complete separation of tasks in well-designed applications, in which both the database and the application can work with data each in its original form.

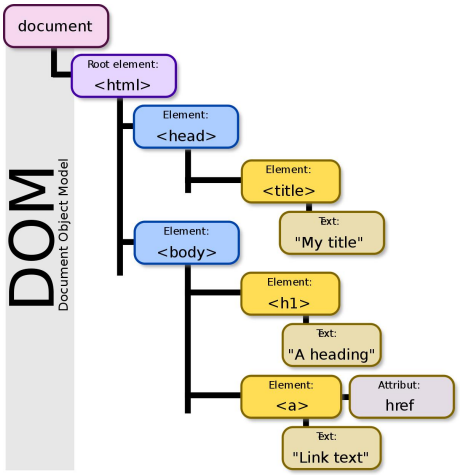

Figure 4. DOM structure

More specifically, the use of ORM solves the problem of the so-called "discrepancy" paradigm, which states that object and relational models do not work very well together. Relational databases represent data in a table format, while object-oriented languages represent them as a connected object graph. The main problems and inconsistencies arise during the storage of this graph of objects in the relational database or its loading:

- The relational model can be much more detailed than the object model. To store one object in a relational database, several tables could be used;
- Relational DBMSs do not have anything similar to inheritance - a natural paradigm of object-oriented programming languages;
- In the DBMS, only one parameter for comparing records is defined: the primary key. While OOP provides both checking the identity of objects and their equality;
- For the connection of objects DBMS uses the concept of foreign keys, in object-oriented languages, the relationship between objects can be only one-way.

The operating principle of ORM.

A key feature of ORM is the mapping that is used to bind the object to its data in the database. ORM creates a "virtual" database schema in memory and allows you to manipulate data already at the object level. This view shows how the object and its properties are associated with one or more tables and their fields in the database. ORM uses the information of this mapping to control the process of converting data between the database and object forms, and to create SQL queries for inserting, updating, and deleting data in response to changes that the application makes to these objects (Fig.3).

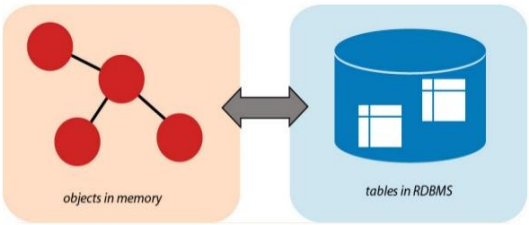

Figure 3. The operating principle of ORM.

Using ORM in the project eliminates the need for the developer to work with SQL and writing a lot of code, often monotonous and error-prone. All the generated ORM code is supposedly well tested and optimized, so you do not need to think about testing it. This is clearly a plus, but at the same time, do not forget about the minuses. The main one is the loss of productivity. This is because most ORMs are designed to handle a wide range of data usage scenarios, much more than any single application will ever be able to use.

4.2. Client side best practices

As it is not difficult to guess, the success of any client part of the application depends on the order in 3 main directions of development: layout, styles, functionality. Each of the items should be described separately.

HTML5 elements.

To provide additional semantic value for our documents, you need to use the appropriate HTML5 elements, such as <header>, <article> and <section>, where necessary. However, in cases where HTML should be as "backward compatible" as possible, you do not need to apply identifiers or classes to them, because older browsers do not understand these elements by default and will not apply the style to them.

Attribute values

You need to use quotation marks to surround all attribute values in HTML, even though quotes are optional in HTML5. This supports consistency between attribute values that contain spaces and those that do not.

Ids vs Classes

HTML elements can be identified using the id and class attributes. An identifier is a unique name for this particular element; no other element on the page should use the same identifier. This uniqueness allows <label> elements to bind themselves to a specific input and URLs to jump to a specific scroll position on the page. Classes are not unique. The same class can be used for multiple elements within a page, and one element can have more than one class in the list with space separators. When names for an identifier or class appear, we use semantic names such as "secondary-nav" or "primary button" that describe the meaning of the component, rather than names like "left-nav" or "blue-button" that describe what an element looks like that can change with time. We also use lowercase names with dashes (kebab-case), unlike camelCase, which is widely used in the development environment.

CSS

CSS is an unusual language that can easily lead to an increase in the amount of code, inconsistencies in the design or conflicting elements of the code. CSS should: be easy to maintain, follow fairly clear templates so that it can be understood, used on many devices.

Tools

Libraries or tools should be selected based on the benefits they provide. Common types of tools related to CSS can include: concatenation of files (placing in bandles), preprocessors, minifactors and postprocessors.

They should be seen in the context of the rest of the site building process, internal and continuous integration processes. Also worth mentioning are css-frameworks like Bootstrap, Semantic - which allow you to place components on the page, making the layout adaptive.

Basic rules for formatting CSS files: use a new line for every selector and every declaration, use a single space before the opening brace in a set of rules, use lowercase for elements and shorthand hex values, e.g., #aaa, hyphenate class selector names; avoid underscores and camelCase, quote attribute values in selectors, use one level of indentation for each declaration, the closing brace of declaration goes in the same column as the first character of the set of rules, use a single blank line between sets of rules.

Putting each selector on its own line and each property on its own line is great for readability and so version control systems can clearly show which parts have changed in a diff.

The attributes within a selector can be alphabetized for easy scanning and so that compression algorithms like gzip have a greater chance of finding repeatable patterns.

Javascript

Today the development of single-page applications is difficult to imagine without Angular.js.

To begin with, Angular.js has a huge community. It includes both members of the permanent development team, and just those who want to contribute to the development of the open source framework. Angular.js hosts a lot of conferences, it is talked about in Hakaton and disputes in the thematic IT community. There are many books and online resources on Angular.js for developers. For customers this means: by choosing Angular.js, you will not only be in a trend, but you can always find developers to support your project.

Declarative way.

When creating templates in Angular.js, a declarative programming paradigm is used. This makes the code more lightweight, makes it easier to read and support, since the required final result is described, and not all steps to achieve it.

The language of the templates in Angular.js is HTML. It is expanded with directives that add information about the required behavior to the code (for example, about the need to download a specific module immediately after the page is loaded). Directives allow you to concentrate on developing logic and work more productively. They can be reused, which also increases the readability of the code.

AngularJS uses an MVC schema that separates the logic, presentation, and application data.

This allows you to create single page Web applications (Single Page Application). In Angular.js, there is a $ http service that provides interaction with remote HTTP servers using XMLHttpRequest or JSONP. When you pass a JavaScript object to the server, it will automatically be converted to a JSON string. After receiving the response, the service will also try to convert the received JSON string to JavaScript. Using the $ http service, you can create your own service with complete control over the processing of URLs and data.

Templates in Angular.js are HTML code, supplemented with elements and attributes of Angular.js. The framework complements the template with information from the model to show the user a dynamic page. To process data and format values obtained from the model, filters are used. They allow you to display the necessary data to the user without having to make changes to the original data.

5. Practical part

Most of the principles and conceptions from this article were applied to “Science to metrics” project, for his improvement and simplifying code realization.

First version of application was released in 2016, and now now has undergone tremendous changes. First of all: client-side part will be turned from dynamic JSP templating to full SPA(Single page application) on Angular platform. The latest Angular version provide strong opportunities of TypeScript, which nowadays considered “Java-like” language in client-side development. Also scss-preprocessor, which simplify writing style sheets for each component in Angular environment without redundancy. Node packet manager used to bind dependencies from other open sources, and Webpack - for minifying and packing static content and sources.

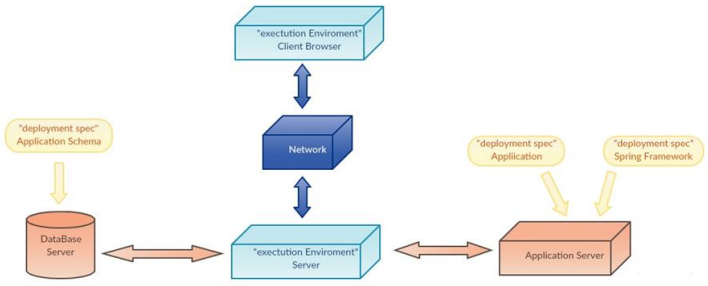

Figure 5. Client-server interaction.

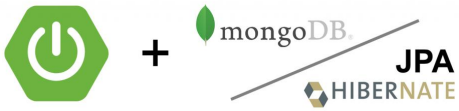

Figure 6. Main back-end components.

For back-end development Spring Boot will be used, a lightweight framework that takes most of the work out configuring Spring-based applications. It will allow to draw more attention to business logic of application instead of configuring project and all of it’s dependencies and also simplifies deploying it. Relational RDBMS MySQL will be changed to NoSQL MongoDB. Decision to change DB approach was based on two main cons, which are valuable to any modern application::

1. Unlike relational databases, NoSQL databases easily store and combine any type of data, both structured and unstructured. You can also dynamically update the schema to evolve with changing requirements and without any interruption or downtime to application.
2. NoSQL databases scale out on low cost, commodity hardware, allowing for almost unlimited growth.

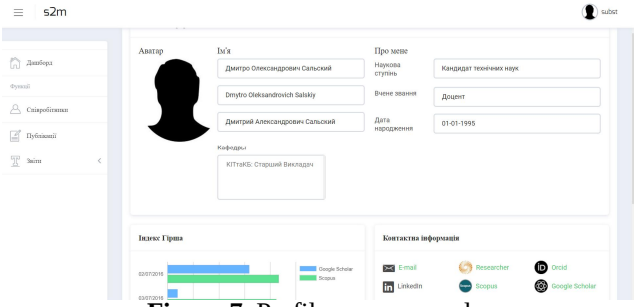

Figure 7. Profile page example.

Results

The result of the development is a software product that should provide access to information regarding the publications of the ONAFT employee and his personal information regarding his publication activities. Many useful functional abilities - like reports generations or data modification will be implemented in this portal app.

Referenses

[1] “Inversion of Control Containers and the Dependency Injection pattern,” martinfowler.com.
[2] J. Weiskotten , “Dependency Injection and Testable Objects,” *Dr. Dobbs Journal*, 2007. [Online]. Available: http://www. ddj. com/development-tools/185300375. [Accessed: 2017].
[3] W. Scott , “Mapping Objects to Relational Databases: O/R Mapping In Detail,” *Agile data*. [Online]. Available: http://www.agiledata.org/essays/mappingObjects.html. [Accessed: 12ADAD].
[4] L. Stevens and R. Owen, “The Truth About a Basic HTML5 Web Page,” *The Truth About HTML5*, pp. 13–15, 2013.
[5] N. Murray, A. Lerner, F. Coury , and C. Taborda, ng-book 2: The Complete Book on Angular 2. Fullstack.io, 2016.
[6] J. Duckett, HTML and CSS: design and build websites. Indianopolis, *IN: Wiley & Sons*, 2011.

Література

[1] Inversion of Control Containers and the Dependency Injection pattern [Електронний ресурс] / – Режим доступу до ресурсу: martinfowler.com.
[2] Weiskotten J. Dependency Injection and Testable Objects [Електронний ресурс] / J. Weiskotten // Dr. Dobbs Journal. – 2007. – Режим доступу до ресурсу: http://www. ddj. com/development-tools/185300375.
[3] Scott W. Mapping Objects to Relational Databases: O/R Mapping In Detail [Електронний ресурс] / W. Scott // Agile data – Режим доступу до ресурсу: http://www.agiledata.org/essays/mappingObjects.html.
[4] Stevens L. The Truth About a Basic HTML5 Web Page / L. Stevens, R. Owen. // The Truth About HTML5. – 2013. – С. 13–15.
[5] Ng-book 2: The Complete Book on Angular 2 / N.Murray, A. Lerner, F. Coury, C. Taborda., 2016. – 626 с.
[6] Duckett J. HTML and CSS: design and build websites. / J. Duckett. – Indianopolis: Wiley & Sons, 2011.

УДК 681.51

РАЗРАБОТКА АЛГОРИТМА УПРАВЛЕНИЯ СИСТЕМОЙ ОТОПЛЕНИЯ И ГОРЯЧЕГО ВОДОСНАБЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ВОЗОБНОВЛЯЕМЫХ ИСТОЧНИКОВ ЭНЕРГИИ

А.А. Процишен[1], **Е.О. Улицкая**[2]

[1,2] Одесский национальный политехнический университет, Украина
ORCID: [1]0000-0001-6241-3458; [2]0000-0002-8572-538X

DOI: 10.15673/atbp.v10i4.818

Аннотация: В исследовании проведен анализ системы отопления и горячего водоснабжения с использованием возобновляемых источников энергии.