LOOK ACADEMIC PUBLISHERS
OPEN ACCESS

INTERNATIONAL JOURNAL OF ENVIRONMENTAL & SCIENCE EDUCATION
2016, VOL. 11, NO. 10, 3861-3883

# Development of Distributed System for Electronic Business Based on Java-Technologies

Aliya Zh. Kintonova[a], Bakkaisha Z. Andassova[a], Madina A. Ermaganbetova[a] and Elmira K. Maikibaeva[a]

[a]L. N. Gumilyov Eurasian National University, Astana, KAZAKHSTAN

**ABSTRACT**

This paper describes the results of studies on the peculiarities of distributed information systems, and the research of distributed systems technology development. The paper also provides the results of the analysis of E-business market research in the Republic of Kazakhstan. The article briefly describes the implementation of a possible solution to a distributed system based on Java-technologies. The topicality of the work is determined by the need of E-business development to increase economic investments, as well as improving the competitiveness of Kazakhstan in the world trade. The paper gives a brief description of the types of E-business, the concept of a distributed system, types of distributed software systems, distributed systems architecture analysis, logical software layers of distributed systems, as well as the analysis of distributed systems technology development and Java-technologies for the development of distributed systems. In addition, an example of a distributed system development in E-business is provided – a Java-based online store, as well as the description of the implementation of a distributed system, the software part and a hardware part of a distributed system, the description of the logical architecture for the online store, the structure and the operation of the system. Also, the name and address of the implemented distributed system are given, which is the result of a study of distributed systems and the implementation of Java-based distributed systems. The development of E-business requires the implementation of an information system for the promotion of goods and services. Today, E-businesses mostly use information systems with distributed structure. The implementation of Java-based distributed systems in E-business is one way of the E-business development in Kazakhstan. The results are applicable to the post-Soviet and developing countries, which include Kazakhstan, as the theoretical basis for the development of E-business.

**KEYWORDS**
Distributed systems, E-business, online store, distributed systems development technologies, Java-technologies

**ARTICLE HISTORY**
Received 17 May 2016
Revised 29 June 2016
Accepted 30 June 2016

## Introduction

In today's society, E-business is a widespread phenomenon. Digital infrastructures are increasingly becoming the necessary basis for E-business around the globe (Camarinha-Matos, Afsarmanesh & Rabelo, 2013). E-businesses are characterized by ever-increasing supply possibilities, ever-increasing global competition and the expectations of consumers. In many ways, the demand for integration directions can be explained by the transition of

**CORRESPONDENCE** Aliya Zh. Kintonova ✉ Aliya_kint@mail.ru

companies and organizations to digital provision of services and the development of E-business (Andam, 2014).

The main types of E-business: online stores, corporate sites, catalogs, ratings, search engines, content projects, web information business, financial services, advertising, trading platforms, e-auctions (Hass, Bichler & Guler, 2013).

The analysis of E-business in Kazakhstan considers the prospects of E-commerce market growth ninefold; thereby increases the need to study this issue, as an essential link for the development of the technological and economic potential (Galiev, 2012).

E-business opens up fresh prospects, for example, now anyone can conduct their business from any part of the world, needing only a computer and Internet access.

The development of an E-business requires the implementation of an effective information system of product promotion (Nikabadi & Jafarian, 2015). The implementation of Java-based distributed E-business systems is one way of E-business development in Kazakhstan. Distributed system is a collection of interacting software components. Each of these components can be regarded as a program module (application), executed in a separate process.

The main purpose is to facilitate the access to remote resources, and to control the sharing of those resources (computers, files, data in the database). To solve this primary problem, a distributed system must meet the following requirements: transparency, openness, flexibility, scalability (Distributed information systems, 2013). Often the task of increasing the overall system capacity by workload distribution of its constituent units acts as the motive for the creation of distributed systems.

## Literature review

At the initial stage of system modeling, attention is paid not only to the division into parts of the system (decomposition), but also to the interaction of these parts (modules) that may be executed on different nodes. The choice of architecture in most cases creates a particular method of decomposition of the problem, as well as ways of interaction (Verissimo & Rodrigues, 2012). A well-designed system structure should provide reliability, manageability, flexibility, and thus allow to build cost-effective solution. In describing the architecture, separate component features are usually not stated; important is the placement of components (including network topology, which can be crucial in the selection of a particular architecture), data distribution methods (and control), workload distribution between the components, as well as ways of interaction (Rozanski & Woods, 2012). The interaction of components is a key aspect for distributed systems. Often the choice of system architecture is dictated by the interaction characteristics. Architecture can be understood as a kind of template, some general idea of decomposition and interaction of the components, the solution, which was used many times before. In a sense, this is an attempt to compensate for the increasing complexity of the system. Another way to overcome the difficulty lies within the model of separation the developed system's layers. The idea is that a lot of problems arise on a constant basis, so the developers need to create a set of ready-made software solutions, general enough to use them in various cases. In this method, application is considered as the top layer, which

uses the functionality of the underlying layer, which is often referred to as middleware (Rosales, Torres & Ramos, 2015). The two bottom layers are often referred to in a general term – a platform. Middleware is the intermediate software between the platform and the actual components of a distributed application. Generally speaking, this level is not mandatory, but its presence is highly desirable. Its purpose is to conceal (disguise) the heterogeneity of platforms and to provide a convenient programming model (Methods and ways of developing distributed systems using Java, 2014).

In some information systems, the use of "thick client" architecture is technologically driven. This may be due to the large amounts of information, the complexity of local information processing, using specialized software, etc. Unfortunately, currently not many high-quality software products have high fault tolerance and productivity. Often, due to the incompetent organization of the system, functioning problems arise, as well as faults in reports. Therefore, the problem of Java-based distributed E-business systems requires a detailed study.

There are such basic approaches to distributed computing systems: the methods of remote procedure calls, multi-layer client-server systems, multi-agent systems, peer-to-peer computing technology and service-oriented approach (Radchenko, 2012).

The paper provides a detailed consideration of the World Wide Web, the development of which created a sharp increase in the interest in distributed systems (Methods and ways of developing distributed systems using Java, 2014), the basic principles and concepts of which are: communication, processes, synchronization, integrity and replication, fault tolerance and security. (Tanenbaum & Van Steen, 2003; Mikov & Zamyatina, 2007).

The three main technologies that support the concept of distributed object systems are the EJB, CORBA and DCOM (Distributed systems review, 2016; Karpov, 2007; Kintonova & Bigalieva, 2015).

Software plays an important role as an e-commerce platform, used to conduct e-auctions, allowing to expand the range of the economies of various countries (Hass, Bichler & Guler, 2013).

EJB technology greatly simplifies the development and configuration of enterprise-level system written in Java. Enterprise Java Beans defines a certain set of multi-purpose and reusable components, called Enterprise beans. When creating a distributed system, its business logic is implemented at the level of these components (Vershinin & Ivanova, 2003).

Java 2, Enterprise Edition (J2EE) is one of the most powerful platforms for the development of modern software projects. There is an inextricable connection between the process of practical development of enterprise software

systems and the principles of object-oriented design (Vershinin & Ivanova, 2003).

JavaBeans has all the functionality of Java programs, as well as the ease of use and rapid progress in the market; it successfully supports the component model for all Java developers (Harold, 1999).

The .NET Remoting is the brand-new technology for building distributed applications, which allows you to build fault tolerant, scalable, secure, fast and easy in maintenance and administration distributed Internet applications, designed for developers who have experience working with Microsoft, .NET Framework and C# (McLean, Naftel & Williams, 2003).

The main purpose of distributed processing is the facilitation of access to remote resources, and the control of shared use of those resources (computers, files, data in databases) (Distributed information systems, 2013).

## *Purpose of the study*

1. To analyze the state of E-business in Kazakhstan, distributed systems architectures, the implementation of distributed systems technology.

2. The implementation of a Java-based distributed system.

### *Research questions*

What types of distributed software systems there are?

What types of distributed systems architectures there are?

### Methods

As a methodological basis were used such general scientific principles of studying the distribution system architectures, as analysis, dialectical and systematic approaches, which allowed to consider the studied phenomena and development processes, identify the strengths and weaknesses, using the comparative approach. In the paper were also applied the techniques and methods of statistical research, the objective design and project-based method, based on which the Java-based website was developed.

Theoretical research basis consists of the studies of domestic and foreign scientists.

### Data, Analysis, and Results

Distributed systems. Distributed system is a set of independent computers, appearing to the users as a single unified system (Tanenbaum & Van Steen, 2003). A distributed system is a software and hardware system, aimed at solving certain tasks. On the one hand, each computing node is an independent element. On the other hand, the software component of a distributed system must provide users with an appearance of a single computer system. Consequently, a distributed system can be characterized by the following characteristics:

– ability to work with different types of devices;

– with a variety of devices suppliers;

– with different operating systems;

– with a variety of hardware platforms (Harold, 1999).

There are such types of distributed software systems: single-stage architecture (created under the influence of the first computer systems, where all three layers (presentation, business logic and data) were parts of one program); two-tiered architecture was developed after the appearance of personal computers (presentation layer was not integrated into other layers); three-tier architecture became even more complex and diverse than client/server architecture, every layer is separated from the other. The presentation layer is placed in a client in a two-tier architecture, business logic is placed in the middle tier, and is referred to as the service layer or an intermediate software layer. Data access layer is located on the third tier and consists of all the servers, integrated together during the architectural decision (Distributed systems review, 2016; Karpov, 2007). From the viewpoint of resource management subsystem, programs, running in the business logic layer, are simply clients, as shown in Figure 1.

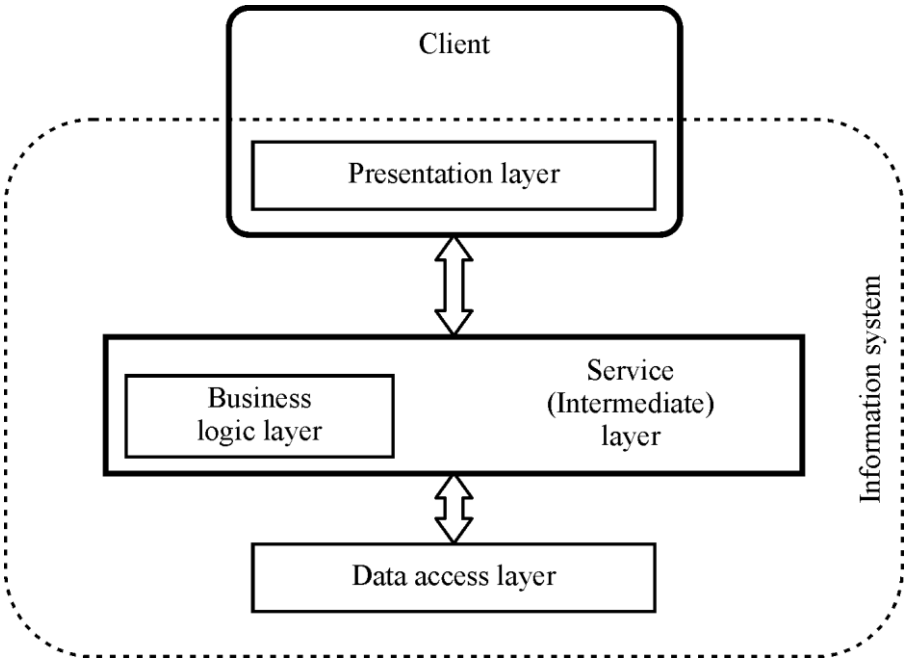


**Figure 1.** Program layers of distributed software

*Logic software layers of distributed systems.* Conceptual distributed systems are built in layers: presentation layer, business logic layer and data access layer. These layers may be only abstractions, existing only at the design stage, but can be clearly seen in software, when implemented in the form of separate subsystems, sometimes using of various tools.

*Presentation layer*. All distributed systems need to communicate with the outside world, with users, programmers, or other systems. Much of this communication is connected with the transformation of information and its presentation to the outside world, making requests and receiving responses. The components of a distributed system, providing these activities, form the presentation layer.

Often, this layer is called a distributed system client. All distributed systems have clients, i.e. entities, using the services, provided by these systems. Clients may be completely external to the system and independent of them. In this case, they are not the presentation layer of the systems themselves. The best examples of programs, designed in this way, are network navigators, processing documents, written in HTML. Presentation layer of a distributed system in this case will be a network server, as well as modules, creating HTML-documents.

Sometimes the client and the presentation layer are merged together. This is typical of client/server systems, in which program functions as the client and the presentation layer at the same time.

*The layer of application logic*. Any software system does not only show information, but also processes data. This processing is carried out by the software, executing the actual operation requested by the client through the presentation layer. This layer is called business logic layer. Sometimes it is referred to as the services, offered by distributed systems. Depending on the complexity of the logic, this layer may be referred to as a business process, business logic, business rules, or simply server.

*Data access layer*. Every software system needs data. It can be stored in databases, file systems and other repositories.

Data access layer programs combine all these elements. Sometimes, to show the layer is executed using a database management system, this layer is called the data layer. This approach has limitations because it focuses only on the data management aspect. However, all external systems, supplying information, need management. This includes not only the database, but also other distributed systems with their layers of presentation, business logic and resource management. Thus, it is possible to build distributed systems recursively, consisting of other systems as components.

The three described layers are the *conceptual structure*s, logically sharing the functionality of most distributed systems. In practical realizations they can be combined in various ways (Karpov, 2007).

*The architecture of distributed systems*. The main characteristic of the interaction between distributed subsystems is their synchronous or asynchronous behaviour. Regarding the studied systems, it would be more accurate to name it blocking or non-blocking interaction. In blocking interaction, parties must wait until the end of the interaction before proceeding to perform next tasks. It should be understood that simultaneous operation of systems' fragments is not related to asynchronous behaviour and non-blocking interaction. Server, receiving a request from one of its clients, can process the request simultaneously with its other clients, while synchronous behaviour deals with client request processing and the server process, handling it.

The client and server parts of a distributed system interact via RPC completely independent from each other, in particular from the used programming languages. In traditional systems, if an application, written in one language, calls a procedure, written in another language, the developer needs to know many technical details of such calls: details of the data types representation in different languages, ways of aligning elements and fill the voids in complex structures. Difficulties would have increased many times, if the interaction between programs, not only written in different languages, but also operating on different computers, was organized this way. All these problems can be eliminated by the remote procedure call (RPC) model. Developers became able to build distributed applications without changing language or programming paradigms.

The basic distributed architecture is a "client-server" model. Client process, running on a user's behalf, requests a service from a server by sending a request and waiting for a response. The server executes some services, such as file system or database service and responds to client requests. The client-server interaction is also known as the request-response execution option (Methods and ways of developing distributed systems using Java, 2014). One server usually has many clients. Client-server applications are also called two-tier, because the client communicates with the server directly (Distributed information systems, 2013). Typically, two-tier architecture can be easily realized, but have scalability problems. Therefore, it is recommended to separate the system into three levels: presentation layer (user interface), business level (service level), and data access layer (see Figure 2).

Presentation layer is realized in clients. It contains programs for the user's interaction with the application.

Processing level can also contain business logic, verifying the correctness of the data, transmitted by client, and processes them according to business rules. This processing may require interaction with data access level or perform local computations. If everything is well, the intermediate level usually passes the results to data access layer for storing and returns the results to the client.

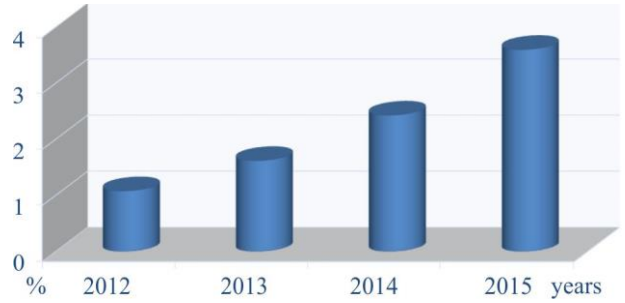


**Figure 2.** The analysis and predictions on business development in Kazakhstan

There are three types of distributed systems architectures:

– the client/server architecture. In this model, a system can be represented as a set of services provided by servers to clients. In such systems, servers and clients are very different from each other;

– the three-tier architecture. In this model, a server provides clients with services not directly, but through a business logic server;

– the distributed objects architecture. In this case, there is no difference between clients and servers, so a system can be represented as a set of interacting objects, whose location does not matter; no difference between a service provider and users.

### *Distributed systems design technologies*

Currently, three different technologies support the concept of distributed object systems. These are EJB, CORBA and DCOM (Distributed systems review, 2016; Hass, Bichler & Guler, 2013).

With its easy to use Java-model, EJB is the easiest and fastest way to build distributed systems. EJB is a good choice for creating RAD-components and small applications in Java. Surely, EJB is not as powerful as DCOM or CORBA. Therefore, the RMI part in the creation of large scalable industrial systems is reduced (Distributed systems review, 2016; Hass, Bichler & Guler, 2013).

Distributed Component Object Model (DCOM) is a software architecture, developed by Microsoft for the distribution of applications among multiple computers in a network. A software component on one machine can use DCOM to send a message (RPC) to a component on another machine. DCOM automatically establishes a connection, sends a message and returns a response to the remote component. Distributed Component Object Model (DCOM) is built into the operating systems Windows NT 4.0, Windows 98 and higher (Distributed systems review, 2016; Hass, Bichler & Guler, 2013).

CORBA specifies the infrastructure of interaction between components (objects) on the presentation level and the business level of the OSI model. It allows us to consider all applications in a distributed system as objects. At the same time, objects can play the role of both a client and server: client, if the object is the initiator of calling the other object's method; server, if another object calls any of his methods. Server objects are typically referred to as "the realization of objects". In practice, most of the objects play the role of both clients and servers at the same time, calling methods on other objects and responding to the outside calls. Thus, by using CORBA it is possible to build a much more flexible system than the client-server system, based on the two-tier and three-tier architecture (Distributed systems review, 2016; Hass, Bichler & Guler, 2013).

The system, built on the distributed object technology, consists of a set of components (objects), interacting with each other. Objects usually are spread over the network and executed separately from each other.

Java.net API, RMI, CORBA, web-services, JMS are the most popular Java-technologies of distributed applications development. These technologies and software are either included into a J2SE distribution kit, or can be freely downloaded from the software companies' websites.

Java.net API provides the ability to implement the network communication using one of two transport protocols: UDP and TCP.

CORBA means Common Object Request Broker Architecture. CORBA is a technology for building distributed systems, standardized and supported by a consortium of OMG. This technology provides a mechanism to describe the interface specifications of interacting systems (assuming that they can be implemented in various programming languages, using a variety of platforms and operating in different network infrastructures), but also provides some important services - such as naming services, events, transaction, security, etc. It allows programs, written in different languages and working in different network nodes, communicate with one another as easily as if they were in the address space of the same process (Tanenbaum & Van Steen, 2003; Karpov, 2007).

Web-services are the advance in the distributed system technologies. The specification of Open Net Environment (ONE) by Sun Microsystems and the initiative of .Net by Microsoft provide the infrastructure for building and deploying web-services. Currently, there are several definitions of a web-service. A web-service can be any application having access to the web, for example, a web-page with dynamic content. In a narrower sense, web-service is an application, providing an open interface, suitable for use by other applications on the web. The specification of ONE requires web-service to be available through HTTP or other web-protocol, to make the exchange of information possible through XML-messages and that they can be found through special search services. To access a web-service a special protocol was developed – Simple Object Access Protocol (SOAP), which is a XML-based means of interaction for many web-service. Web-services are especially attractive due to the high degree of compatibility between various systems, that they provide.

JMS (Java Messaging System) is an interface to external systems, aimed at work through messages. JMS is an “old” technology - its first specification was published in 1998. Currently javax.jms package includes jdk, while Sun Application Server implements the JMS support as one of the services.

In the development of JMS, the primary task was to create generic Java API for message-oriented application programming, and ensuring the independence from specific implementations of the corresponding messaging services.

Thus, the program, written using the JMS, will work correctly with any messaging system that supports the specification (or having appropriate interfaces).

Since JMS is a shell, or an interface, describing the methods available for an application, applications will need a specific implementation of JMS

interfaces, called a JMS provider. They are created by independent developers, so now there is a variety of such implementations (including, for example, the implementation included into Sun Application Server and distributed together with the J2EE, as well as MQSeries by IBM, JMS WebLogic service by BEA, SonicMQ by Progress etc.).

RMI technology (Remote Method Invocation) is the next stage in the development of RPC (its object implementation). When accessing the RMI, client uses a link, containing the network address of a server and the full path to the object on the server, including the local object identifier in the address space of the server. Also, the link is encoded in the protocol stack, used for the client-server interaction. RMI (Remote Method Invocation, i.e. call a remote method), integrated with JDK1.1, is a product of JavaSoft and implements a distributed computing model. RMI allows client and server applications to invoke methods through a network of clients/servers running on the Java Virtual Machine. Although RMI is considered lightweight and less powerful than CORBA and DCOM, still, it has a number of unique features, such as distributed, automatic management of objects and the ability to send the objects from machine to machine. Object is a standard designed software module, containing data and operations on that data. The data contained in the object are called properties, and the operations on the data are called methods. The object's methods can be accessed through the interface, provided by software systems. An object can implement multiple interfaces. In turn, the interface description may also have multiple objects. Distributed object is an object, whose interface is on a different machine than the object. A characteristic feature of distributed objects is that their data is not shared. They are stored on the same machine. On other machines only those interfaces are available, which are implemented by the object. When a client accesses the distributed object, the control is passed to the program interface object implementation, similar to a client stub called proxy. Proxy transits parameters from the calling program to the client computer operating system and back, like a client stub. On the server side, the parameter transit from an OS server to the methods of the object and back is carried out by an analogue of server stub program, called skeleton. Distributed system objects exist in the forms, accepted in a particular programming language (Distributed information systems, 2013).

RPC model is the core of most distributed systems, according to Figure 3. Many later models are based on this model, such as Remote Method Invocation (RMI) and database stored procedures. Often RPC model is used as a low-level entity for the implementation of more complex forms of interaction.

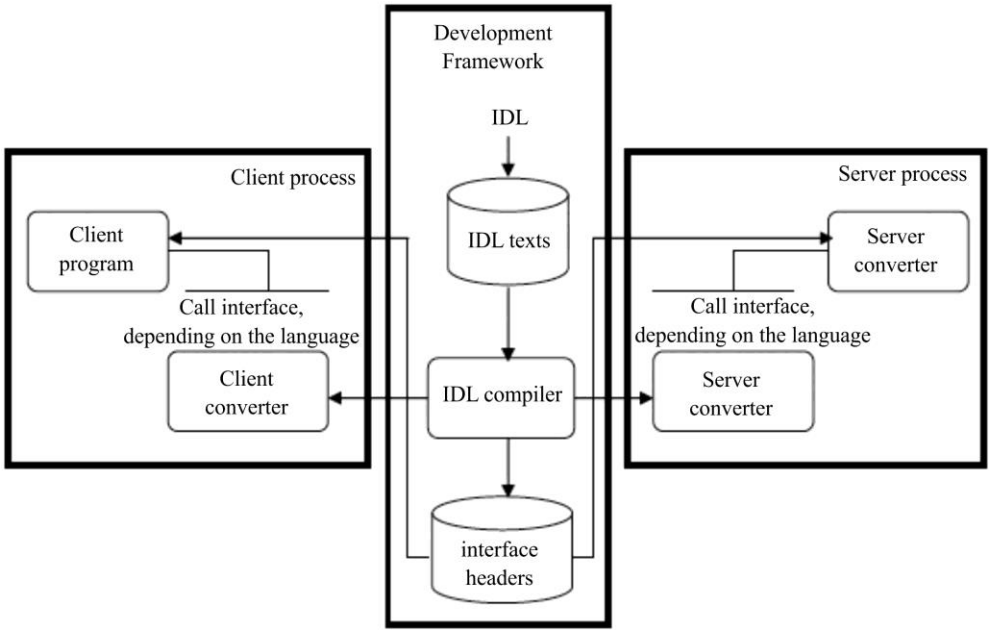


**Figure 3.** The development of distributed applications, using RPC

RPC process is performed as follows:

– The process on the client device calls the procedure on the server device;

– The client process is suspended;

– The process execution of the called procedure starts on the server device;

– The result is transmitted to the client device;

– The client process is resumed.

While implementing the RPC model, it is necessary to do some premastering. The first step should be to determine the procedure interface, using the Interface Definition Language (IDL). This definition makes a brief description of the procedure parameters, passed to it before the work and returned after work. Procedure Definition in IDL can be regarded as a service specification provided by the server.

The second premastering step is the translation of the created description. Any implementation of a RPC mechanism, any integration platform, using RPC or a similar concept, contains a special interface compiler. The results of translation are:

– Client Converter. Each procedure's header description in IDL file results in the creation of a separate converter for the procedure. Converter is a program, adding itself to the client application after translating. The structure

of a client converter includes server-searching software, data formatting, interaction with the server, getting and transmitting a response in the form of returned parameters of a client process.

– Server Converter is similar to the client, but it implements the server side of a call. It consists of programs, receiving a request from a client, formatting data, calling a real procedure, implemented on a server, and returning the results to a client. Like the client converter, after the translation, server adapter add itself to the server program. A server program, including the implementation of a procedure, called by a remote client, does not have to be written in the same language that was used to implement the client. It makes it possible to achieve a significant increase in cross-language interoperability, i.e. the possibility to carry out the interaction of programs, written in different programming languages.

– Software Templates and links. IDL and its compiler help in the development of procedures, creating support files. Many modern compilers generate program templates for servers, such as programs containing procedure headers, so the developer has to create only the implementation of these procedures (Distributed information systems, 2013).

*Such Java technologies can be used in the development of distributed systems:*

– Java Foundation Classes (Swing) (JFC).

– JavaHelp.

– Java Native Interface (JNI).

– Java Platform Debugger Architecture (JPDA).

– Java 2D API.

– Java Web Start.

– Certification Path API.

– Java Database Connectivity (JDBC).

– Java Advanced Imaging (JAI).

– Java Authentication and Authorization Service (JAAS).

– Java Cryptography Extension (JCE).

– Java Data Objects (JDO).

– Java Management Extensions (JMX).

– Java Media Framework (JMF).

– Java Naming and Directory Interface (JNDI).

– Java Secure Socket Extensions (JSSE).

– Java Speech API (JSAPI).

– Java 3D.

– Metadata Facility.

– Java Content Repository API.

## *The implementation of a Java-based distributed system*

As a result of the study of distributed systems based on e-business, an online-shop was developed, which is already in trial. The online-shop is put on the Internet at sandy.kz.

Online-shop structure should ensure the following basic principles:

– centralized approach for information processing;

– avoided duplication of input data and its increased reliability by identifying a previously input information;

– a mechanism of user access rights differentiation;

– ensured conflict-free expansion of functions, conflict-free expansion of users, working with the store.

Physically, online store consists of two parts:

Software:

– a database on SQL Server database platform;

– a set of software modules;

– a set of software modules that provide the ability to download the reports and upload the external data.

Hardware:

– hardware-software system to accommodate the database server;

– hardware-software system to accommodate the application server.

"Sunday" online store home page. Figure 4 shows the home page of the online store.

Registration page. Figure 5 shows the online store's registration page.


**Figure 4.** Online-shop's home page

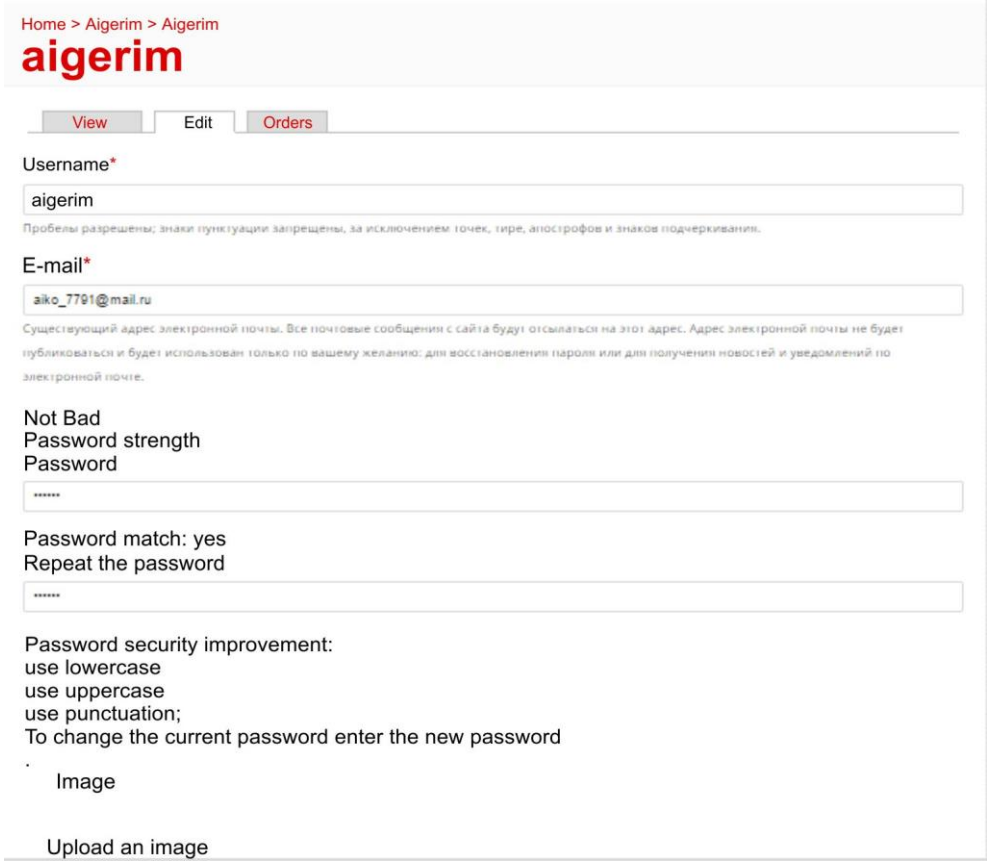


**Figure 5.** Online store's registration page

The moderator data window. Figure 6 shows the moderator data window:

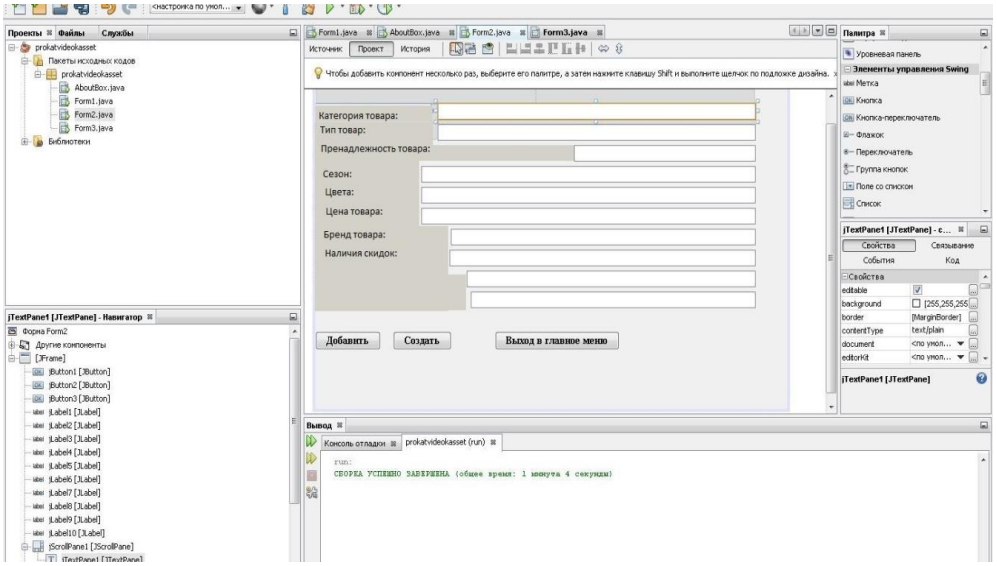


**Figure 6.** Online store registration page

According to Figure 7, the logical architecture of the online-shop has 3 levels:

– data access level – provides the storage and access to data systems, transaction management;

– business logic level – provides data transfer between the presentation layer and the data access, performs the operation, based on the user's input data;

– presentation layer – provides interfaces for displaying and inputting information, processes user's commands and then converts them into operation on the business logic level.

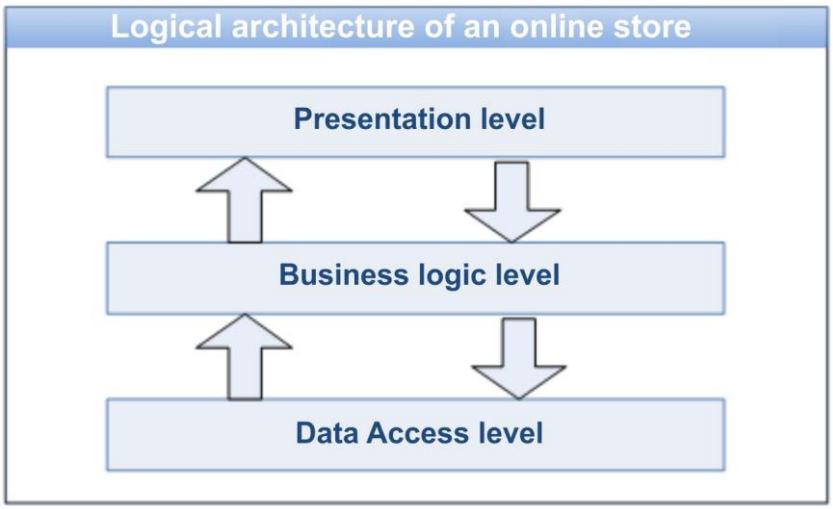


**Figure 7.** Logical architecture of an online store

Based on the logical architecture, the complete online store architecture, providing its distribution, was built, as indicated on Figure 8.

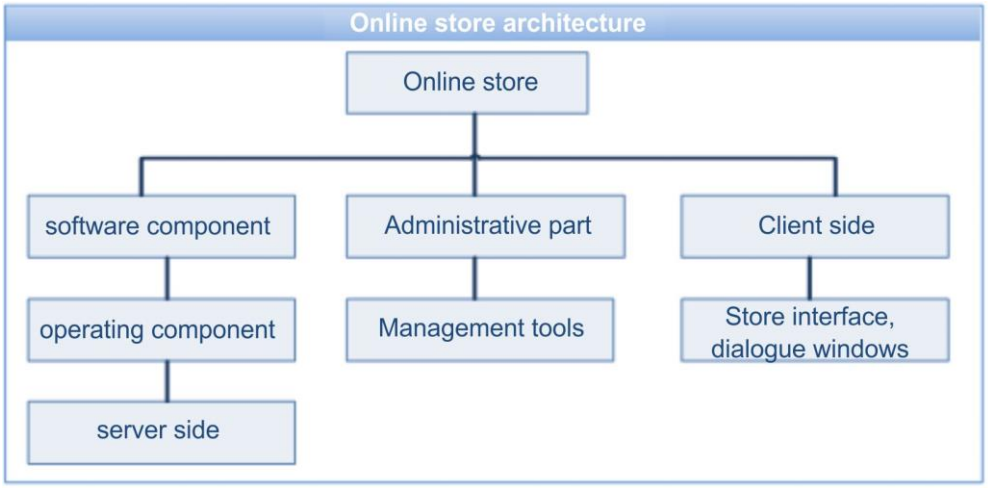


**Figure 8.** Online store architecture

The structure and operation of the system. Based on architectural solution, the following modules (subsystems) were determined:

1. The “Frontend” module;
2. The “Backend” module;
3. The “Data storage” module;

1. “Frontend” is aimed at providing a single point of entry into online store for its users. The module must be a site and should provide all interested persons with information about the sale of goods. It should provide a registered user with the opportunity to work in a personal account and perform the following functions:

– Logging in;

– Editing personal data;

– Ordering goods;

– Moving goods into the basket;

– Order status check.

2. "Backend" is aimed at user feedback. The module is an administrative part, providing the following functions:

– Creating and filling sections / subsections of the site menu;

– Adding information about a product;

– Adding product to cart and ordering.

3. "Data Storage" provides the storage of information on the Internet within the store sections. Through the business logic users and site administrators call the module with a request.

Their basic roles and functions are given in the Table 1.

**Table 1.** Roles and their functions in the system

| *Roles* | *Functions* |
|---|---|
| Guest | Catalogue view<br>Registration |
| Registered user | Logging in<br>Personal data editing<br>Ordering a product<br>Putting the product into the basket<br>Order status check |
| Store moderator | Creating and filling the site menu sections/subsections<br>Adding the product information<br>Adding a product into the basket and order finalization<br>Order status change |

Business processes used in the system. The basic task of online store is the order of goods and their processing. Based on this data, core business processes, used in the system, were developed.

1) Signing up to an online store

To register in a store, user must complete the registration procedure, according to the following sequence (Figure 9):

– click the "Register" button in the registration window;

– enter the login information and e-mail address, which will be tied to a user account in the future;

– the registration confirmation is sent to the email address of the user, which was entered in the registration form;

– pass a registration confirmation so the user is automatically added to the account in the system.

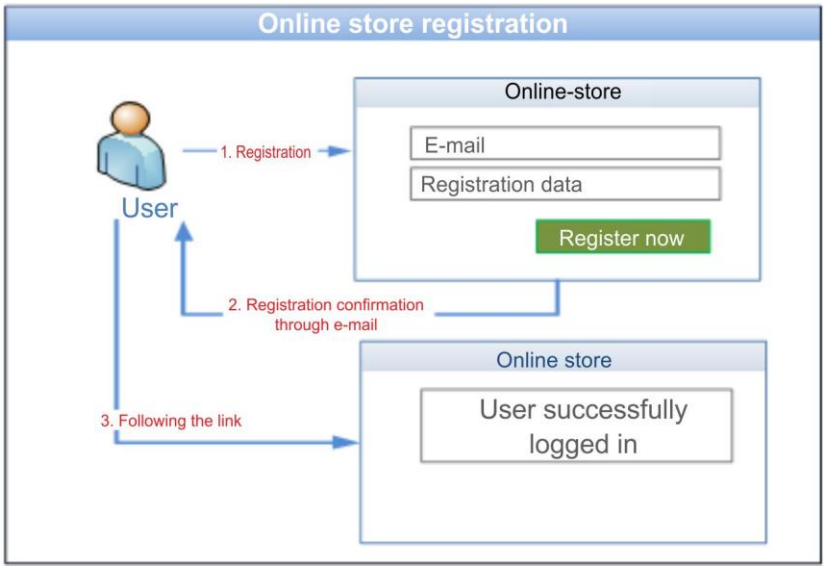


**Figure 9.** Online store registration

2) Ordering a product

To order from an online store you must first log in, and then perform the following steps:

– Select the desired item;

– Add selected items to the cart;

– Order the product, which is in the cart;

– Confirm the order through a notification e-mail, as shown in Figure 10.


**Figure 10.** Ordering a product in an online-store

3. Fill in the content of an online store:

The content of an online store is filled by system administrators. To do this, the administrator must select the relevant sections and to add information on the goods.

4. Processing the order:

After ordering goods, the order is processed by an administrator. Administrator checks the goods and contacts the user. Administrator has a workstation for processing the orders.

## *The technologies used in the development of the online shop system*

The Java tools for the distributed system development of "Sunday" online store:

– Java Foundation Classes (Swing) (JFC) is a set of Java-classes libraries, used to create graphical user interfaces and the implementation of other graphical features in the Java-client applications. Guide Introduction to Swing (EN), as well as articles on Design of dynamic interfaces in Swing (EN) and Improving access and graphical user interfaces (EN) will help you get started with Swing.

– JavaHelp is a platform-independent extensible help system that enables developers and technical writers to embed applets in the manual pages, software components, applications, operating systems and devices, as well as help create Web-systems. Read Creation of help systems for Java-based applications (EN).

– Java Platform Debugger Architecture (JPDA) is a debugging infrastructure in Java SE.

– Java 2D API consists of a set of classes to work with two-dimensional graphics, providing the opportunities to create images and manage alpha channels, classes for manipulating color palettes and color conversion, as well as operators to work with images (see. Guide Introduction to Java 2D (EN)).

– Java Web Start technology simplifies the deployment of Java-based applications, enabling users to download and run a feature-rich software tools, such as spreadsheets, in one mouse click, without installation.

– Java Database Connectivity (JDBC) is an API, providing the means to access most relational data sources from Java-based applications. Through it one can connect to a variety of SQL databases and other tabular data sources, such as spreadsheets and flat files (Hass, Bichler & Guler, 2013).

– Java Advanced Imaging (JAI) is an object-oriented API, which is a simple high-level programming model that simplifies the manipulation of images.

– Java Authentication and Authorization Service (JAAS) is a technology that provides a service means to authenticate users and verify their access rights. It includes Java-implementation of a standard PAM (Pluggable Authentication Module) infrastructure and supports authentication at the user level.

– Java Data Objects (JDO) is a standard abstract model of long-term storage of the Java-objects, based on interfaces. With it, developers can directly save copies of Java-classes in permanent storage (e.g. database). This model in some cases can replace a direct entry in the file, serialization, JDBC, and the use of server-side EJB components, both managed by containers (Container Managed Persistence - CMP) and self-storing the state (Bean Managed Persistence - BMP).

– Java Management Extensions (JMX) package provides tools for building distributed, modular, dynamic and Web-accessible applications for managing and monitoring devices, software and networks, based on providing the services.

– Java Media Framework (JMF) allows you to add audio, video and other media information into applications and applets in Java.

– Java Naming and Directory Interface (JNDI) is a unified interface to access various services and directory names within a corporate network. With it, the application can effectively connect to different services and directory names in a heterogeneous enterprise environment.

– Java Secure Socket Extensions (JSSE) is a set of packages, ensuring the secure exchange of information on the Internet.

– Metadata Facility mechanism allows developers to define attributes for classes, interfaces, fields and methods, so that they may be subjected to special treatment on the part of development tools, deployment and libraries at runtime.

– Java Content Repository API is an API to access content repositories in Java SE independently from the implementation. These repositories are high-level information management systems, an advanced version of the classic data repositories etc. (Kintonova & Bigalieva, 2015).

## Discussion and Conclusion

As a basis for any studies in the area of e-commerce growth prospects in Kazakhstan, the predictions made in 2011 by Askar Zhumagaliyev are used. According to him, in 2014 the market will grow to 1.2 billion US dollars. “According to our expectations for 2014, the volume of e-commerce market will reach about $ 1.2 billion, while Internet advertising will grow almost twice”, - Digital Communications Kazakhstan (2011). At the same time, according to A. Zhumagaliev, (2011), e-commerce turnover in Kazakhstan was about 300 million US dollars, the volume of online advertising – about 6 million US dollars, i.e. the growth of approximately 4 times in 2-3 years.

At the same time, the CEO at Processing.kz K. Gorozhankin, (2014) gives more optimistic predictions. According to him, by 2015, e-commerce market in Kazakhstan will increase by nine times. At the moment the market of e-commerce in Kazakhstan is about 0.45% of the total market. The estimates growth are as follows: in 2013 the share of e-commerce will account for 1.8%, in 2014 - 2.7%, and in 2015 figure will reach 4% by the time the e-commerce market will reach 3.6 billion US dollars. Vice-Minister of Transport and Communications of Kazakhstan S. Sarsenov (2016) is less optimistic – he estimates the e-commerce market in Kazakhstan will reach $4 billion only by 2020. It is possible that he takes into account the risks that are hovering over the world economy as a danger of “second coming” of the crisis that threatens to

overshadow the previous one. Figure 1, based on the data, stated above, provides an analysis and predicts the business development in Kazakhstan.

Still, there is other, more modest evaluation of existing realities. In particular, according to a survey conducted by the "Chocolife.me" analytical research center, the entire e-commerce market in Kazakhstan in 2011 is worth over 133 million dollars. Basically, it is created by "AirAstana" and other companies, selling airplane and train tickets, the company said. Regarding payment methods, in this case 78.36% of buyers pay with bank cards, 18.31% with cash and 3.33% pay for the services via terminals. Leading position in terms of revenue took the following online retailers: «Disti» - 5,7 million, «Sulpak» - 3,7 million dollars, «Alser» - 3,6 million US dollars (Galiev, 2012).

Kazakhstan experiences only the first wave of e-commerce boom, so it is necessary to achieve a minimum level of spending on the online services (including the correlation of average salaries in Kazakhstan) to the level of 60-70 dollars per 1 inhabitant in the coming years, which this is about 11 billion US dollars (Galiev, 2012).

However, when a technology disappears from view, very often it reappears under a new name. The result is a continuous mixing of the basic concepts with the latest approaches toward development.

The development of e-business requires the implementation of an information system for the promotion of goods and services. Today, e-businesses mostly use information systems with distributed structure. The implementation of distributed e-business systems, based on Java-technologies, is one way of e-business development in Kazakhstan.

## Implications and Recommendations

E-business development in Kazakhstan is one of the key tasks, which is confirmed by the "Information Kazakhstan-2020" state program. The choice of e-business technology development will directly affect the dynamics of e-business development in Kazakhstan.

This paper shows the performance of Java-technologies, based on the created online store. Using java technology will increase the investments into e-business, which will contribute to the technological and economic growth.

The advantages of Java technologies:

– writing software on one platform and running it practically on any platform;

– creating programs, running in a Web browser and accessing Web Services;

– development of applications on the server side for online forums, stores, polls, HTML forms processing etc;

– unification of applications or services using Java to create highly specialized applications or services;

– creation of powerful and efficient applications for mobile phones, remote processors, microcontrollers, wireless modules, sensors, gateways, consumer products, and practically any other categories of electronic devices.

The results are applicable to the post-Soviet and developing countries, including Kazakhstan, as the theoretical basis for the development of e-business.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Notes on contributors

**Aliya Zh. Kintonova** is a PhD, Associate Professor of Computer Science and Information Security Department, L. N. Gumilyov Eurasian National University, Astana, Kazakhstan.

**Bakkaisha Z. Andassova** is a PhD, Associate Professor of Computer Science and Information Security Department, L. N. Gumilyov Eurasian National University, Astana, Kazakhstan.

**Madina A. Ermaganbetova** is a PhD, Associate Professor of Computer Science Department, L. N. Gumilyov Eurasian National University, Astana, Kazakhstan.

**Elmira K. Maikibaeva** holds a Master Degree, Senior Lecturer of Computer Science Department, L. N. Gumilyov Eurasian National University, Astana, Kazakhstan.

## References

Andam, Z. R. (2014). *e-Commerce and E-business.* Direct access: https://en.wikibooks.org/wiki/E-Commerce_and_E-Business

Camarinha-Matos, L. M., Afsarmanesh, H., & Rabelo R. (2013). *E-business and Virtual Enterprises: managing business-to-business.* New York: Springer, 245 p.

Distributed information systems (2013). Direct access: http://5fan.ru/wievjob.php?id=48479

Distributed systems review. (2016). Direct access: http://www.uran.donetsk.ua/~masters/2008/fvti/prihodko/library/dist2.htm

Galiev, A. (2012). *E-commerce in Kazakhstan: a true boom?* Direct access: http://www.computerworld.kz/articlekz/3646/.

Gorozhankin, K. (2014). Who helps the e-commerce in Kazakhstan make money. *Forbes Kazakhstan, 37,* 44-46

Harold, A. (1999). *JavaBeans.* Moscow: Lori, 163 p.

Hass, Ch., Bichler, M., & Guler K. (2013) Optimization-based decision support for scenario analysis in electronic sourcing markets with volume discounts. *Electronic Commerce Research and Applications, 2(3),* 152-165.

Karpov, L. E. (2007). *Software distributed systems architecture.* Moscow: MAX Press, 132 p.

Kintonova, A. Zh. & Bigalieva, A. (2015). The implementation technologies of distributed systems. *Science and Education" digest at the 10th International Scientific conference of Students and young Researchers* (pp. 1096-1100). Astana.

McLean, S., Haftel, J., Williams, K. (2003). *Microsoft .NET Remoting.* Moscow: "Russian Editorial" Publishing, 384 p.

Methods and ways of developing distributed systems using Java. (2014). Direct access: http://old.mkgt.ru/files/materials/878/lecture1.pdf

Mikov, A. I. & Zamyatina, E. B. (2007). *Distributed systems and algorithms. Lecture course.* Direct access: http://window.edu.ru/resource/466/57466

Nikabadi, M. S., & Jafarian, A. (2015). Effect of Necessary Factors for Deploying E-Business Models on Business Performance and Supply Chain Performance in Auto Industry. *Technological Solutions for Sustainable Business Practice in Asia, 1,* 153.

Radchenko, G. I. (2012). *Distributed computing systems.* Chelyabinsk: Mir, 226 p.

Rosales, J. H., Torres, G., & Ramos, M. (2015). A Middleware for Integrating Cognitive Architectures. *Brain Informatics and Health: 8th International Conference,* BIH 2015, London, 353-357.

Rozanski, N., & Woods, E. (2012). *Software systems architecture: working with stakeholders using viewpoints and perspectives.* London: Addison-Wesley, 256 p.

Sarsenov, S. (2016). *Informational Kazakhstan-2020.* Direct access: http://strategy2050.kz/ru/page/gosprog4

Tanenbaum, A. & Van Steen, M. (2003). *Distributed systems. Principles and paradigms*. St. Petersburg: Peter, 877 p.

Verissimo, P., & Rodrigues, L. (2012). Distributed systems for system architects. *Springer Science & Business Media*, *1*, 22-34.

Vershinin, M. M., & Ivanova, E. B. (2003). *Java 2, Enterprise Edition. Design and development technologies.* St. Petersburg: BHV-Petersburgh, 317 p.

Zhumagaliev, A. (2011). *Digital Communications Kazakhstan.* Deirect access: http://digitalcom2011.kz/