



Universitatea Tehnică a Moldovei

**OPENMONEY – PUBLIC ACQUISITION PROCESSING
OPENMONEY – PROCESAREA ACHIZITIILOR PUBLICE**

Student:

Plăcintă Alexandru

Conducător:

dr. conf. Ciorbă Dumitru

Chișinău 2017

Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Admis la susținere

Șef departament: dr. conf.univ. Ciorbă D.

„_____” 2017

Openmoney – public acquisition processing
Openmoney – procesarea achizițiilor publice

Proiect de licență

Student:_____ (A.Placinta)

Conducător:_____ (D. Ciorba)

Consultanți:_____ (R.Melnic)

_____ (G. Covdii)

Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică
Specialitatea Tehnologii Informaționale

Aprob
dr.conf.univ. Dumitru Ciorbă
șef departament

„__” _____ 2016

CAIET DE SARCINI
pentru proiectul de licență al studentului

_____ *Plăcinta Alexandru*

(numele și prenumele studentului)

1. Tema proiectului de licență *Openmoney – public acquisition processing /*
Openmoney – procesarea achizițiilor publice

confirmată prin hotărârea Consiliului facultății de la „ 24 ” *octombrie* 2016

2. Termenul limită de prezentare a proiectului 31.05.2017

3. Date inițiale pentru elaborarea proiectului *Sarcina pentru elaborarea*
proiectului de diplomă.

4. Conținutul memoriului explicativ

Introducere

- 1. Descrierea și analiza domeniului de studiu*
- 2. Analiza aplicației din punctul de vedere arhitectural*
- 3. Implementarea și tehnologii de dezvoltare*
- 4. Argumentarea economică*

Concluzii

5. Conținutul părții grafice a proiectului

Diagrame Use-Case, de activitate, bază de date și deployment a sistemului. Interfețele
funcționalităților de bază ale sistemului.

6. Lista consultanților:

Consultant	Capitol	Confirmarea realizării activității	
		Semnătura consultantului (data)	Semnătura studentului (data)
<i>G. Covdii</i>	<i>Argumentarea economică</i>		
<i>R. Melnic</i>	<i>Controlul calității Standarde</i>		

7. Data înmânării caietului de sarcini 15.09.2016

Conducător Ciorba Dumitru

semnătura

Sarcina a fost luată pentru a fi executată

de către studentul Placinta Alexandru 15.09.2016

semnătura, data

PLAN CALENDARISTIC

Nr. crt.	Denumirea etapelor de proiectare	Termenul de realizare a etapelor	Nota
1	<i>Elaborarea sarcinii, primirea datelor pentru sarcină</i>	<i>15.09.16– 15.10.16</i>	<i>10%</i>
2	<i>Studierea literaturii de domeniu</i>	<i>16.10.16– 16.11.16</i>	<i>20%</i>
3	<i>Alegerea și pregătirea de lucru a softului</i>	<i>17.11.16 – 17.12.16</i>	<i>20%</i>
4	<i>Realizarea programului</i>	<i>18.12.17 – 30.04.17</i>	<i>25%</i>
5	<i>Descrierea programului, diagramele UML</i>	<i>01.05.17 – 15.05.17</i>	<i>10%</i>
6	<i>Testarea aplicației</i>	<i>16.05.17– 28.05.17</i>	<i>10%</i>
7	<i>Finisarea proiectului</i>	<i>29.05.17– 31.05.17</i>	<i>5%</i>

Student Plăcinta Alexandru ()

Conducător de proiect Ciorba Dumitru ()

DECLARAȚIA STUDENTULUI

Subsemnatul Plăcinta Alexandru, declar pe proprie răspundere că lucrarea de față este rezultatul muncii mele, pe baza propriilor cercetări și pe baza informațiilor obținute din surse care au fost citate și indicate, conform normelor etice, în note și în bibliografie. Declar că lucrarea nu a mai fost prezentată sub această formă la nici o instituție de învățământ superior în vederea obținerii unui grad sau titlu științific ori didactic.

Semnătura autorului _____

Semnătura conducătorului de licență _____

Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică și Microelectronică
Catedra Automatică și Tehnologii Informaționale

AVIZ

la proiectul de licență

Tema „Openmoney” procesarea achizițiilor publice

Studentul Placinta Alexandru gr. FAF-131

1. **Actualitatea temei:** se exprima prin faptul ca oamenii simpli vor avea modalitate simpla si clara de a vizualiza cheltuielile de bani de catre diferite institutii(suma de bani, pentru ce servicii, care companii au cistigat tendere respective s.a.m.d) sau de a vedea informatie detaliata despre niste entitati, atit cit si jurnalisti profesioniști in investigatii vor economisi timp utilizind acest instrument si vor facilita de niste functionalitati aditionale de analiza.

2. **Caracteristica tezei de licență:** Lucrarea a fost efectuată în strictă conformitate cu exigențele impuse. La elaborarea tezei de licență a fost corect identificată necesitatea acestui instrument pentru oameni simpli si jurnalisti de investigatii prin faptul expunerii informatiei in mod mai clar si cautarea mai rapida decit folosind exceluri.

3. **Estimarea rezultatelor obținute:** Activitate de cercetare independentă a studentului s-a soldat cu realizarea obiectivelor propuse, cu implementarea corectă a prototipului grație analizei profunde a problemei.

4. **Corectitudinea materialului expus:** Materialul expus în proiectul vizat corespunde cu consecutivitatea logică de expunere și abordare a tematicii alese, la fel o concordanță absolută dintre materialul teoretic și partea aplicativă.

5. **Calitatea materialului grafic:** Expunerea materialului grafic corespunde consecutivității materialului expus, modalitatea aleasă permite ca informația să fie accesibilă perceperii cititorului.

6. **Observații și recomandări:** Dupa importarea datelor din excel in baza de date de tip graf au fost observate erori in linkuirea relatiilor intre entitati si incapacitatea de a diferentia persoane cu acelasi nume, motivul fiind linkuirea pe baza de denumiri a entitatilor din cauza lipsei ID-urilor. Trebuie de gasit solutie pentru a obtine date cu ID, iar referitor la persoane chiar daca legea privind date personale interzice aceasta de cerut pseudo ID asupra celui real doar pentru a putea diferentia persoanele.

7. **Caracteristica studentului și titlul conferit:** Studentul a dat dovadă de cunoștințe profunde în domeniul tehnologiilor informaționale și aptitudini de aplicare a cunoștințelor teoretice precum și a abilităților practice obținute la disciplinele fundamentale. Teza corespunde complet specialității solicitate din acest considerent propun evaluarea studentului _____.

Rezultatele obținute în cadrul tezei îmi permit să recomand admiterea tezei de licență a dlui Placinta Alexandru spre susținere și să o apreciez cu nota _____.

Conducătorul

tezei de licență *lector superior, magistru*

Dumitru CIORBĂ

29 mai 2017

Rezumat

Transparenta este un punct important in contextul societatii, de aceea am decis sa realizez un produs in sfera achizitiilor publice. "Openmoney" - proiectul rolul caruia este de arata cheltuiile de bani de catre institutii si informatie detaliata despre fiecare entitate in mod mai clar decit posibilitatea existenta in excel cu posibilitate de a vizualiza grafuri de relatii. Aceasta va fi util cit pentru oameni simpli cit si pentru jurnalisti de investigatii economisind timp sau introducind functionalitati adaugatoare de analiza.

Dupa revizuirea structurii datelor publice deschise oferite de egov.md sa observat ca exista entitati precum companii, persoane (fondatori si proprietari) si institutii. Intre persoane si companii exista relatii (fondare si detinere), exista si intre institutii si companii care sunt achizitii. Acesta este un model perfect pentru reprezentarea a datelor in model de tip graf. In acest fel, putem profita de acest model pentru a procesa intr-un aspect mai general, este posibil sa traversam relatiile, sa gasim cea mai scurta cale intre anumite entitati, cum ar fi fluxul de bani de la institutie la o anumita persoana, sau sa gasim programatic relatii suspecte legate de achizitii specifice, astfel incat exista mai multe posibilitati de a fi efectuate cu aceste date.

In proiect au fost multe decizii din punct de vedere tehnologic, decizia arhitecturii generale, tehnologii si instrumente pentru fiecare parte modul, dar cea mai mare atentie si decizie a fost la alegerea tehnologiei privind procesarea datelor dupa model de tip graf, acel sistem de gestionare a bazelor de date care va indeplini necesitatile si va fi optimizat perfect pentru proiect.

Abstract

Transparency is an important point in the context of society, which is why I decided to make a product in the field of public procurement. "Openmoney" - the role of which is to show the money spent by institutions and detailed information about each entity more clearly than the existing possibility in excel with the possibility of viewing graphs of relationships. This will be useful for both simple people and investigative journalists saving time or introducing additional analysis functionalities.

After reviewing the structure of open government data offered by egov.md was noticed that there are entities like companies, persons (founders and owners) and institutions. Between persons and companies exist relations (founding and ownership), also exist between institution and companies that are acquisitions. This is a perfect model for graph representation of data. In such a way we can take advantage of this model to process in more general aspect, is possible to traverse relations, to find shortest path between some entities like money flow from institution to a specific person, or find programmatic suspicious relationship related to specific acquisitions, so there are much more possibilities to be performed with this data.

In the project there were many decisions in terms of technology, the decision of the general architecture, technologies and tools for each module, but the biggest attention and decision was the choice of the data processing technology according to the graph, the database management system that will meet needs and will be optimized for the project.

Table of contents

List of tables	10
List of figures	11
Listings	12
Introduction	13
1 Field Activity Analysis	14
1.1 IT role in data transparency	14
1.2 Open Government Data	14
1.3 Similar systems	17
2 Modeling and Designing the Information System	19
2.1 Analysis of project	19
2.1.1 Definition of features for simple user and admin	19
2.1.2 Request for shortest paths between person and institution	21
2.1.3 Definition of class entities	22
2.2 Implementation	23
2.2.1 REST backend	25
2.2.2 Client (browser)	33
2.2.3 Used technologies	38
2.3 Database Data Model	43
2.4 Server Environment	44
3 Review of system in action	45
3.1 Home page	45
3.2 Companies	45
3.3 Persons (founders and owners)	49
3.4 Institutions	51
3.5 Connections between person and institution	52
3.6 Complex relation	52
3.7 About	53
4 Economic Evaluation of the Project	54
4.1 Project description	54
4.2 SWOT Analysis	54
4.3 Economic motivation	56

					UTM 526.2.861 ME							
Mod.	Coala	Nr. document	Semnăt.	Data	„OPENMONEY” procesarea achizițiilor publice			Litera	Coala	Coli		
Elaborat	Placinta A.											
Conducător	Ciorbă D.									8	102	
Consultant	Melnic R.							UTM FCIM FAF – 131				
Aprobat	Ciorbă D.											

4.3.1	Tangible and intangible asset expenses	57
4.3.2	Salary expenses	58
4.4	Individual person salary	59
4.4.1	Indirect expenses	60
4.4.2	Wear and depreciation	60
4.4.3	Product cost	61
4.4.4	Economic indicators and results	62
4.5	Economic conclusions	63
5	Platform usage	64
	Conclusions	66
	References	68

					UTM 526.2.861 ME	Coala
<i>Mod</i>	<i>Coala</i>	<i>Nr. document</i>	<i>Semnăt.</i>	<i>Data</i>		9

List of Tables

4.1	Analysis of project using SWOT matrix	55
4.2	Time schedule	56
4.3	Tangible assets expenses	57
4.4	Intangible asset expenses	57
4.5	Direct expenses	58
4.6	Salary expenses	58
4.7	Indirect expenses	60
4.8	Total Product Cost	61

List of Figures

1.1	Information about company (bizzer.md)	17
1.2	History of a company (idno.md)	18
2.1	Admin's actions	20
2.2	User's actions	21
2.3	User requests shortest paths between person and institution	22
2.4	Class diagram	23
2.5	System architecture	24
2.6	Deployment diagram of server environment	44
2.7	Deployment diagram of server environment	45
3.1	Home Page	46
3.2	Companies	46
3.3	Company's main information	48
3.4	Company's relationship structure traversing only in ONE DIRECTION	48
3.5	Company's tenders	49
3.6	Persons	50
3.7	Person's relationship structure with BIDIRECTIONAL traversing	50
3.8	Institutions	51
3.9	Acquisitions of an institution	51
3.10	Connections	52
3.11	Complex relation	53
3.12	Page about project	53
5.1	Part of <u>anticoruptie.md</u> 's article - "Achizițiile CNA // Șase mașini Skoda Rapid, cumpărate cu 1,5 milioane de lei"	64
5.2	Part of <u>anticoruptie.md</u> 's article - "Proprietarii site-urilor de știri și interesele pe care le promovează"	65
5.3	Part of <u>anticoruptie.md</u> 's article - "Moratoriu de ochii lumii. Cine construiește benzina în Capitală"	65

Listings

2.1	Graph processing controller	25
2.2	Graph processing Service	27
2.3	Element Storage DAO	30
2.4	DB context configuration	31
2.5	Company HTML template	33
2.6	Company service to interact with REST	35

Introduction

In a small group it is easy to track the actions of one or another person to keep the whole state in your mind of what is happening. In a such case you don't need extra effort to be sure in your community, but once we are trying to think in term of big society there appears a problem. People try to introduce a abstraction and try to decompose society in different groups with their own responsibility. It is hard to track the actions of such groups and in result they try to expose some reports about what they are doing. But such reports usually are hell for simple human to understand the real situation. So those all need to be processed and to show in terms more clear for human.

Actually the reports are stored in a primitive structure such excel or csv. So my proposal is: first step would be to import those reports of tenders and companies into a more natural representation, and i am saying about a graph database, and the second step would be to facilitate the advantage of graph DB engine to do different kinds of processing from easiest like to visualize deep hierarchy of fondation relationships of a source company to more complicated processing like to try programatically detect posible situations of corruption: for example multiple tenders that are won by different direct companies which may be fonded indirect by the same company, or another example is how money can be stolen from a institution to a person in form of tender through a deep sequence of companies being hard to detect.

Such application can become a tool for simple users to visualize in easy way somehow the state of what is happening and a tool for professional journalists to do a more faster analysis in this domain of tenders.

1 Field Activity Analysis

Data is part of everything we do, especially given the current open data movement. From financial market performance to farmer's market locations, weather to health care, bridge and road safety to population information, significant amounts of data are yielded and available for aggregation and analysis, and can be applied to improve public services. This is the philosophy behind the open data movement—that if we make all of this data available to the public, at least the high-value data, we can crowdsource public service issues and come up with the best possible solutions.

1.1 IT role in data transparency

Some definitions for Data Transparency:

- The ability to easily access and work with data no matter where they are located or what application created them.
- The assurance that data being reported are accurate and are coming from the official source.

Open data is only as good as the data analytics platforms and true data transparency policies on which it relies. Bringing big data, open data and data transparency together empowers data to solve some of the world's most challenging problems.

For researchers, public servants, data scientists and citizens to use data to spot problems and find solutions, they need big data tools to scrub and manage the data sets from their original format so they can perform investigations and operations with them. This is where the policy objectives of open data, government transparency and the technological power of big data come together. In addition, policies need to support the technology requirements. For example, while the current open data and government transparency policies allow for PDF documents to count as open data, in some cases, not all data analytics platforms can read PDF files.

The government transparency and open data movements both have the honorable goal of making data available to anyone who wants it. This isn't just in recognition of the need to better govern, but a real step toward creating a better world. Certainly open data will improve government, but it also empowers citizens and builds economic value, not just by monetizing a resource we already have, but also by all the opportunity created from the intelligence that it enables. While the policies have paved the way, only together with big data technology will open data realize its full potential. Big data and open data are a powerful combination that can make a positive change in this world.

1.2 Open Government Data

Some definitions for Open Government Data:

- Data produced or commissioned by government or government controlled entities
- Data which can be freely used, reused and redistributed by anyone.

Open data is the idea that some data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control. The goals of the open data movement are similar to those of other "open" movements such as open source, open hardware, open content and open access. The philosophy behind open data has been long established (for example in the Mertonian tradition of science), but the term "open data" itself is recent, gaining popularity with the rise of the Internet and World Wide Web.

Reasons for Open Government Data:

- Transparency. In a well-functioning, democratic society citizens need to know what their government is doing. To do that, they must be able freely to access government data and information and to share that information with other citizens. Transparency isn't just about access, it is also about sharing and reuse — often, to understand material it needs to be analyzed and visualized and this requires that the material be open so that it can be freely used and reused.
- Releasing social and commercial value. In a digital age, data is a key resource for social and commercial activities. Everything from finding your local post office to building a search engine requires access to data, much of which is created or held by government. By opening up data, government can help drive the creation of innovative business and services that deliver social and commercial value.
- Participatory Governance. Much of the time citizens are only able to engage with their own governance sporadically — maybe just at an election every 4 or 5 years. By opening up data, citizens are enabled to be much more directly informed and involved in decision-making. This is more than transparency: it's about making a full "read/write" society, not just about knowing what is happening in the process of governance but being able to contribute to it.

Government data shall be considered open if it is made public in a way that complies with the principles below:

- Complete

All public data is made available. Public data is data that is not subject to valid privacy, security or privilege limitations. While non-electronic information resources, such as physical artifacts, are not subject to the Open Government Data principles, it is always encouraged that such resources be made available electronically to the extent feasible.

- Primary

Data is as collected at the source, with the highest possible level of granularity, not in aggregate or modified forms.

If an entity chooses to transform data by aggregation or transcoding for use on an Internet site built for end users, it still has an obligation to make the full-resolution information available in bulk for others to build their own sites with and to preserve the data for posterity.

- Timely

Data is made available as quickly as necessary to preserve the value of the data.

- Accessible

Data is available to the widest range of users for the widest range of purposes.

Data must be made available on the Internet so as to accommodate the widest practical range of users and uses. This means considering how choices in data preparation and publication affect access to the disabled and how it may impact users of a variety of software and hardware platforms. Data must be published with current industry standard protocols and formats, as well as alternative protocols and formats when industry standards impose burdens on wide reuse of the data.

Data is not accessible if it can be retrieved only through navigating web forms, or if automated tools are not permitted to access it because of a robots.txt file, other policy, or technological restrictions.

- Machine processable

Data is reasonably structured to allow automated processing.

The ability for data to be widely used requires that the data be properly encoded. Free-form text is not a substitute for tabular and normalized records. Images of text are not a substitute for the text itself. Sufficient documentation on the data format and meanings of normalized data items must be available to users of the data.

- Non-discriminatory

Data is available to anyone, with no requirement of registration.

Anonymous access to the data must be allowed for public data, including access through anonymous proxies. Data should not be hidden behind “walled gardens.”

- Non-proprietary

Data is available in a format over which no entity has exclusive control.

Proprietary formats add unnecessary restrictions over who can use the data, how it can be used and shared, and whether the data will be usable in the future. While some proprietary formats are nearly ubiquitous, it is nevertheless not acceptable to use only proprietary formats. Likewise, the relevant non-proprietary formats may not reach a wide audience. In these cases, it may be necessary to make the data available in multiple formats.

- License-free

Data is not subject to any copyright, patent, trademark or trade secret regulation. Reasonable privacy, security and privilege restrictions may be allowed.

Because government information is a mix of public records, personal information, copyrighted work, and other non-open data, it is important to be clear about what data is available and

what licensing, terms of service, and legal restrictions apply. Data for which no restrictions apply should be marked clearly as being in the public domain.

1.3 Similar systems

After analyzing the market of data transparency projects in Moldova based on data from egov.md i highlighted two projects: [bizzer.md](#) and [idno.md](#).

These projects together with [openmoney.md](#) have similar functionalities to display basic information about companies with some parameters that are shown in Figure 1.1.

The screenshot shows the 'bizzer' website interface. At the top is a blue navigation bar with the logo and links: 'Căutare entitate', 'Despre noi', and 'Contactați-ne'. The main heading is 'Societatea cu Răspundere Limitată SENS MEDIA'. Below this, there are three main sections:

- General:** A table with company details.

Stat	Moldova
IDNO / Cod	1008600021169
Forma	Societate cu răspundere limitată
Data înregistrării	22.04.2008
Adresa juridică	mun. Chișinău, sec. Bulucani, str. Alba-Iulia, 194/3, ap. 15
- Fondatorilor:** A list of founders.

Novac Pavel
Societatea Cu Răspundere Limitată New Media Grup
- Conducătorilor:** A list of managers.

Novac Pavel

Below these sections are two more panels:

- Genuri de activitate nelicentiate:** A list of unlicensed activity types.

Editarea de programe
Publicitate
Realizarea de programe și consultanța în domeniul dat
- Entități similare:** A list of similar entities.

Societatea cu Răspundere Limitată TIGERCORP
Societatea cu Răspundere Limitată FX TRADE GROUP
Societatea cu Răspundere Limitată SENS MEDIA
Societatea cu Răspundere Limitată SORMACAPITAL
Agenția de Presă INFOMARKET MEDIA Societate cu Răspundere Limitată
Societatea cu Răspundere Limitată TV REPUBLICA
Societatea cu Răspundere Limitată BEST ACCOUNT
Societatea cu Răspundere Limitată CREDITEX & CO

Figure 1.1 – Information about company (bizzer.md)

[idno.md](#) has unique feature showing all history of a company - like changing name or founder-s/owners that is shown in Figure 1.2.

After reviewing the structure of open government data offered by [egov.md](#) i noticed that there are entities like companies, persons (founders and owners) and institutions. Between persons and companies exist relations (founding and ownership), also exist between institution and companies that are acquisitions. This is a perfect model for graph representation of data.

So the key feature that is unique in [OPENMONEY](#) is facilitation of graph theory in domain of open government data.



Cod fiscal/IDNO

1008600021169



Istoricul companiei

*Istoricul companiei este disponibil din noiembrie 2014

Aprilie-2015 Fondatori: **din in** Societatea cu Răspundere
Limitată NEW MEDIA GRUPAugust-2015 Fondatori: **din** Societatea cu Răspundere
Limitată NEW MEDIA GRUP **in** Societatea cu
Răspundere Limitată NEW MEDIA GRUP, NOVAC
PAVEL

Genurile de activitate nelicențiate (3)

[Editarea de programe](#)[Realizarea de programe și consultanța în domeniul dat](#)[Publicitate](#)

Genurile de activitate licențiate (0)

Figure 1.2– History of a company (idno.md)

2 Modeling and Designing the Information System

In this chapter will be described process of analysis, designing and development of the system.

2.1 Analysis of project

UML (Unified Modeling Language) is a standard notation for the modeling of real-world objects as a first step in developing an object-oriented design methodology. Its notation is derived from and unifies the notations of three object-oriented design and analysis methodologies:

- Methodology for describing a set of objects and their relationships
- Object-Modeling Technique
- Approach which includes a use case methodology

Among the concepts of modeling that UML specifies how to describe are: class (of objects), object, association, responsibility, activity, interface, use case, package, sequence, collaboration, and state. In general UML is a graphical language which helps to visualize, specify, construct and document all parts of a system. Before UML software design was inconsistent and hard to share. Software engineers were having problems with explaining and understanding different notations in software design invented by other software architecture. In this way by creating the standarts for different elements That's why UML has become the international language of software development, allowing engineers to exchange their designs freely.

UML uses two main diagrams types to model a system: structural and behavioral. Structure diagrams show the things in a system being modeled. In a more technical term they show different objects in a system. In the structural diagrams enters: composite, deployment, package, profile, class, object and component. Behavioral diagrams shows what should happen in a system. They describe how the objects interact with each other to create a functioning system. In the behavioral diagrams enters: communication, use case, diagram, sequence, timing and interaction. This project is making use of class, sequence and use case diagrams to present most parts of the design

2.1.1 Definition of features for simple user and admin

Use case diagrams specify the events of a system and their flows. They are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. Now when the initial task is complete use case diagrams are modelled to present the outside view. So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.

- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

First of all need to define the interaction of people with the system, the abilities of service and to define this from point of view of ADMIN and USER.

First of all system need some data to be imported to start to work, so we start from defining ADMIN's interaction.

In Figure 2.1 Admin is responsible for:

- prepare XLS files with information of entity to a specific format for system and upload, service will process data and import into Graph DB (OrientDB)
- removing and changing relations or entities which was wrong imported

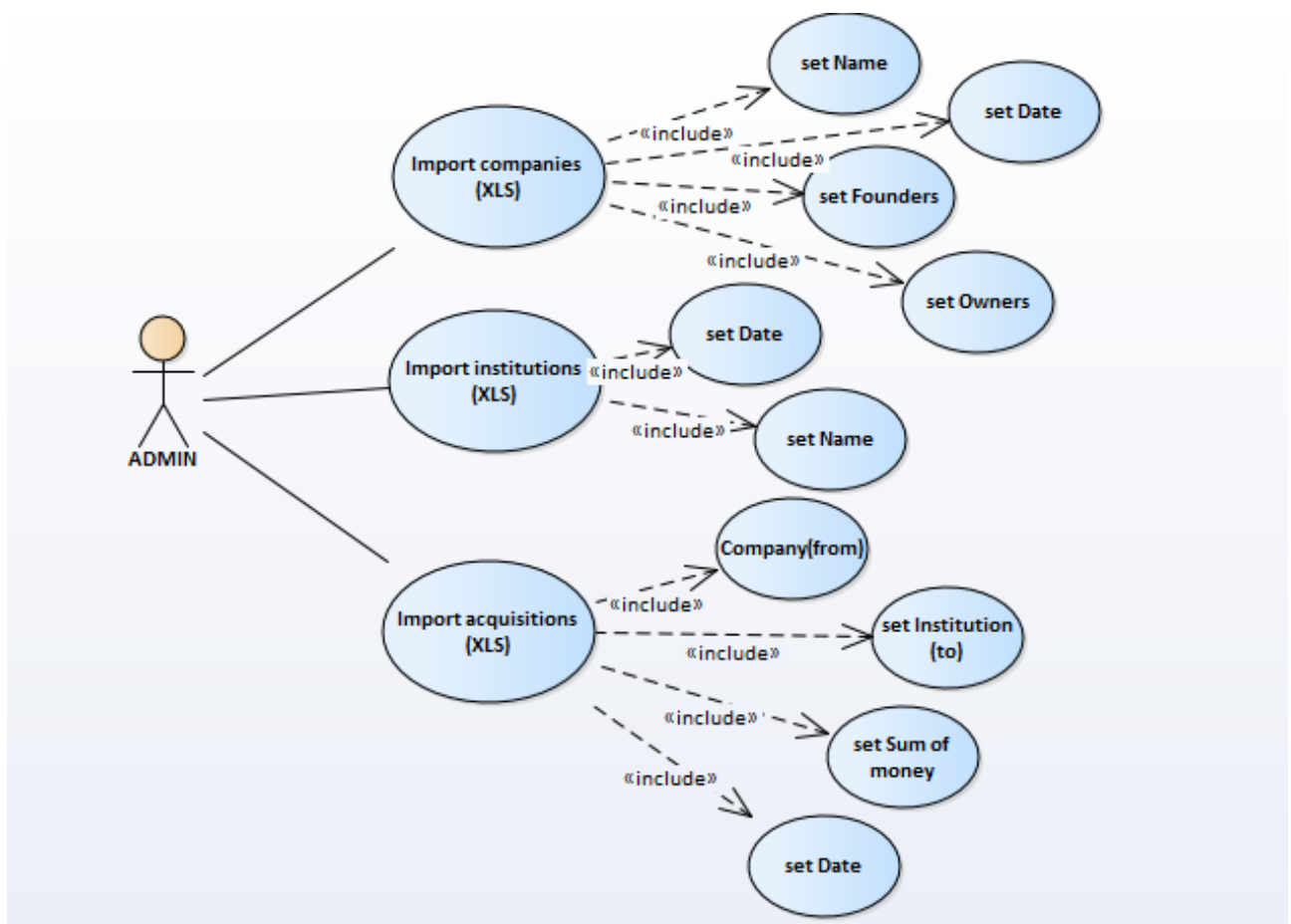


Figure 2.1 – Admin's actions

In Figure 2.2 User can visualize processed data in a well exposed form:

- Check basic info of entity, visualize hierarchy of relations in form of a graph
- View shortest paths between a person and institution (possible ways how money can distinguish from institution to a person)
- View suspicious cases where more companies have acquisitions with the same institution that have also the same founder above in hierarchy of foundation and ownership

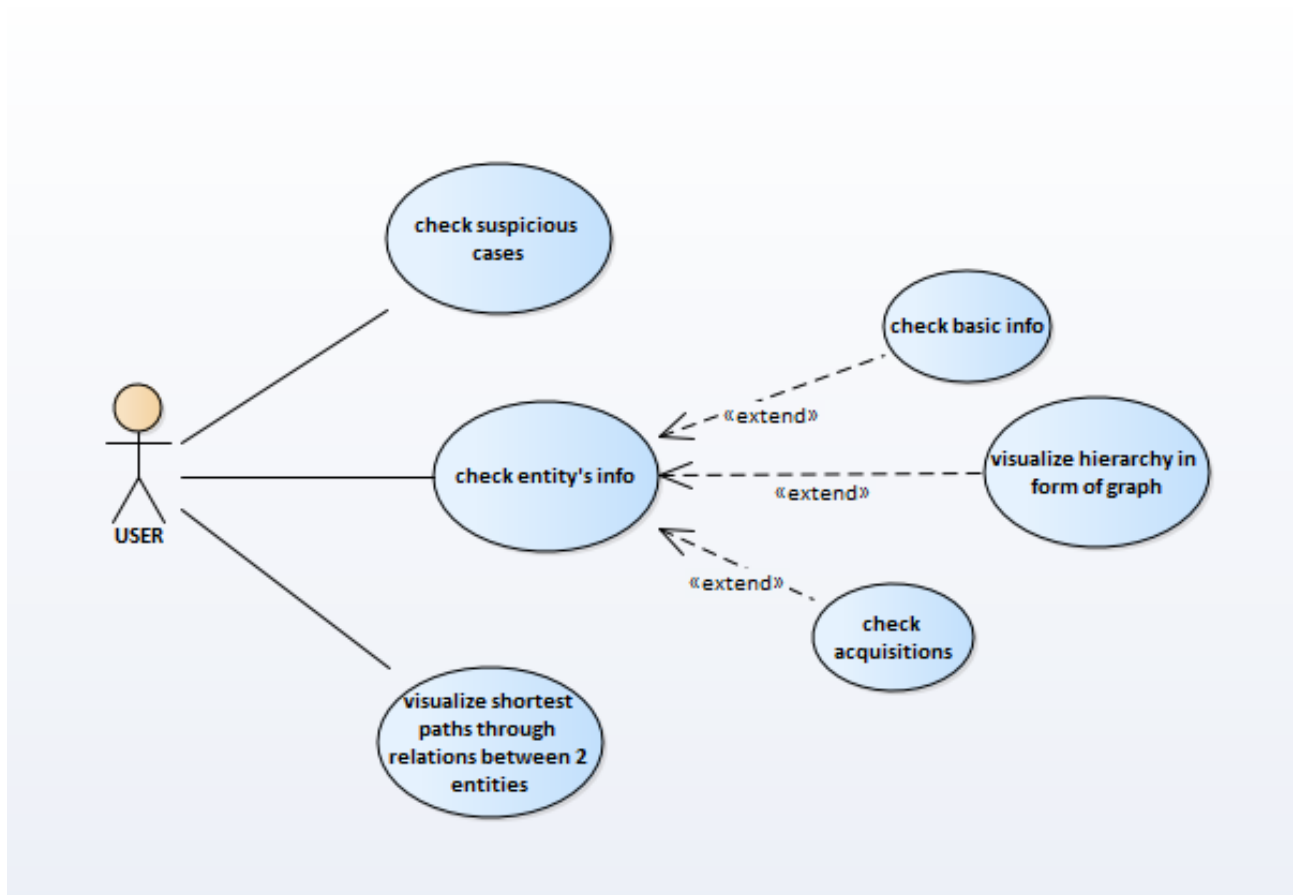


Figure 2.2 – User's actions

2.1.2 Request for shortest paths between person and institution

The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. It is also a form of interaction diagram which shows objects as lifelines running down the page, with their interactions over time represented as messages drawn as arrows from the source lifeline to the target lifeline. Sequence diagrams are good at showing which objects communicate with which other objects, and what messages trigger those communications. Sequence diagrams are not intended for showing complex procedural logic.

In Figure 2.3 is the request for shortest path and how it occurs in time and how is going interaction between components like controller, service for validation of input data and data access object for retrieving the necessary data.

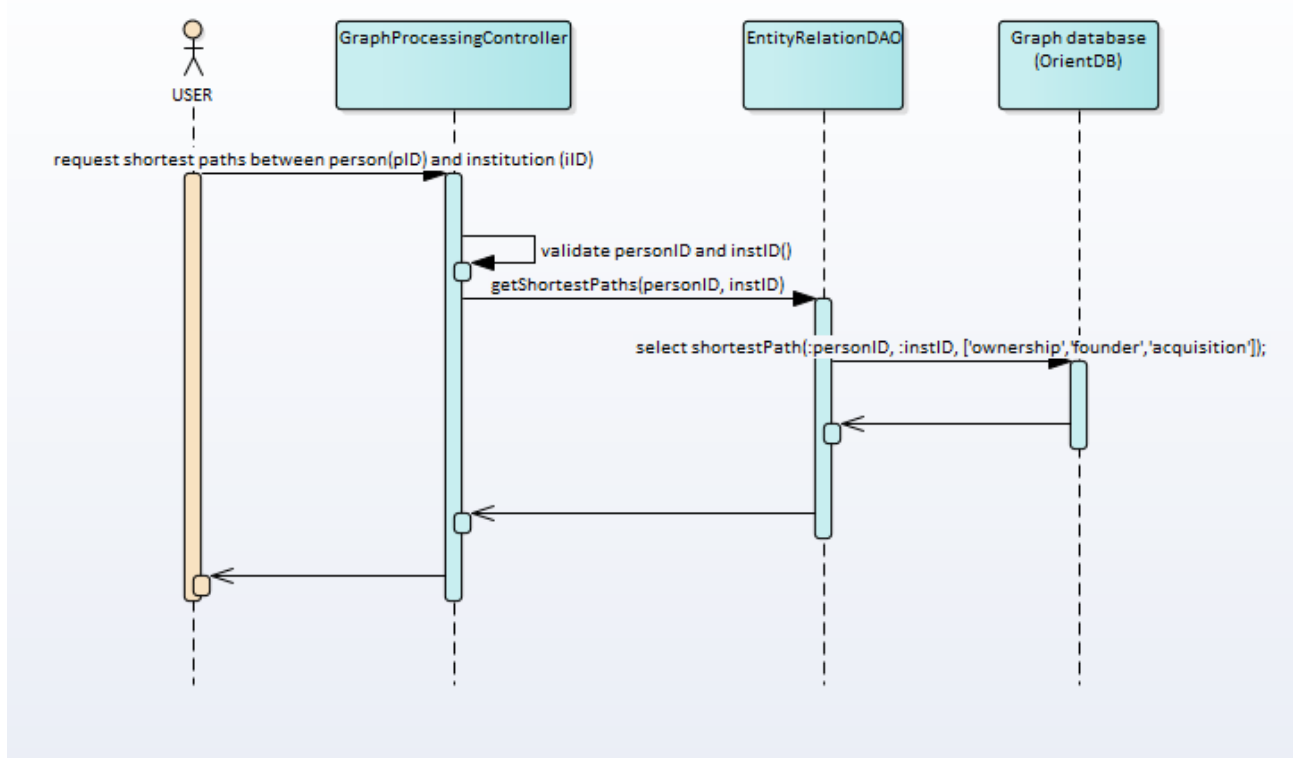


Figure 2.3– User requests shortest paths between person and institution

2.1.3 Definition of class entities

The class diagram shows the building blocks of any object-orientated system. Class diagrams depict a static view of the model, or part of the model, describing what attributes and behavior it has rather than detailing the methods for achieving operations. Class diagrams are most useful in illustrating relationships between classes and interfaces. Generalizations, aggregations, and associations are all valuable in reflecting inheritance, composition or usage, and connections

Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

In Figure 2.4 are represented classes used for graph processing and relationships between them, here we have one class from each layer and some classes as domain model of system.

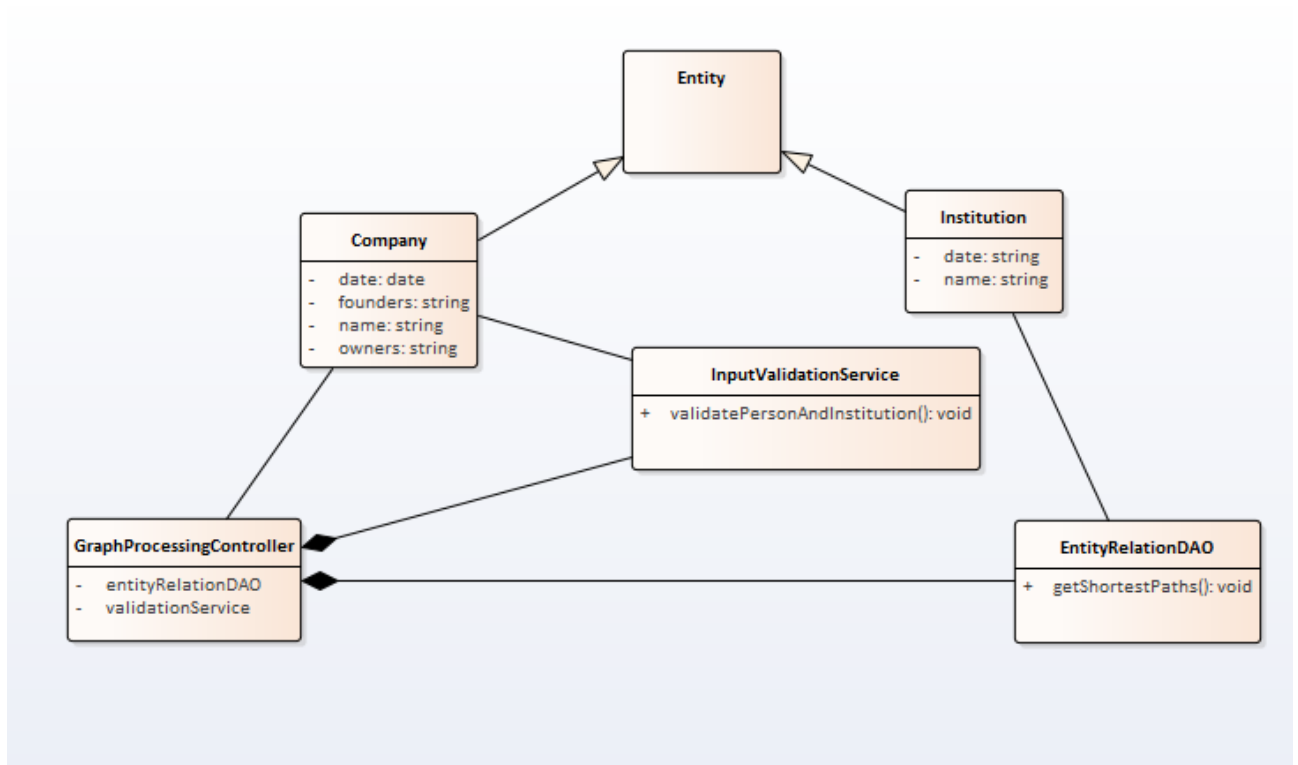


Figure 2.4 – Class diagram

2.2 Implementation

In this subsection will be discussed architecture of system, describing each layer with some code listings and used technologies.

Form-intensive enterprise class applications are ideally suited for being built as single page web apps. The main idea compared to other more traditional server-side architectures is to build the server as a set of stateless reusable REST services, and from an MVC perspective to take the controller out of the backend and move it into the browser.

So we can analyze system from point of view of client and server part separately, check Figure 2.5 for general architecture of system.

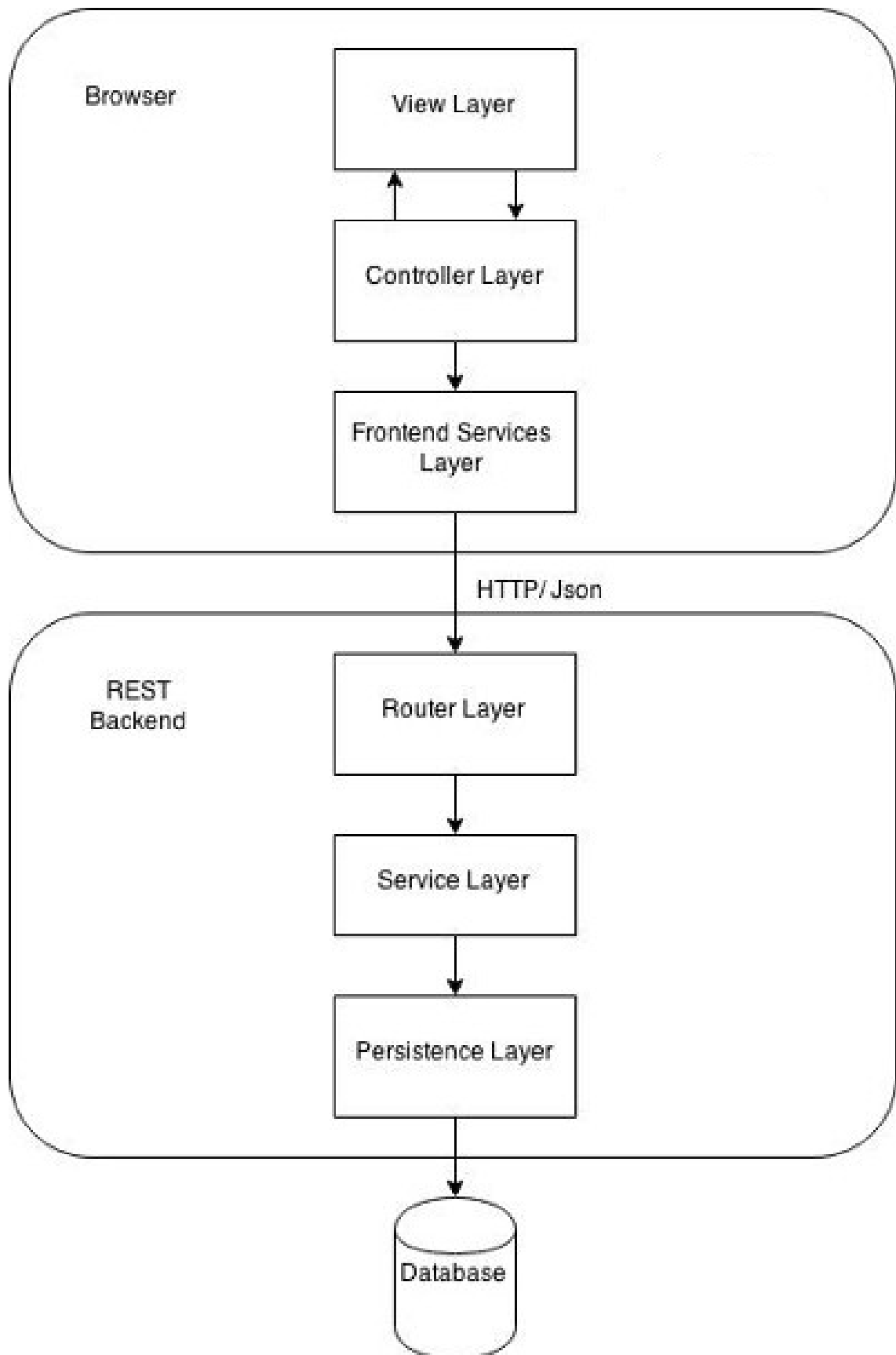


Figure 2.5– System architecture

2.2.1 REST backend

The backend of an enterprise frontend application can be built in a very natural and web-like way as a REST API. The same technology can be used to provide web services to third-party applications - obviating in many cases the need for a separate SOAP web services stack.

From a DDD perspective, the domain model remains on the backend, at the service and persistence layer level. Over the wire only DTOs go by, but not the domain model.

The backend is built using the usual backend layers:

- Router Layer: defines which service entry points correspond to a given HTTP url, and how parameters are to be read from the HTTP request

In Listing 2.1 is controller that is used for validating of parameters from incoming request for getting shortest paths between 2 entities and delegate service components with business logic for processing that stuff.

```
1 package md.openmoney.rest.controller;
2
3
4 import java.util.*;
5
6 import com.tinkerpop.blueprints.impls.orient.OrientGraph;
7 import md.openmoney.rest.dao.ContractStorage;
8 import md.openmoney.rest.dao.ElementStorage;
9 import md.openmoney.rest.dao.InstitutionStorage;
10 import md.openmoney.rest.dto.GraphContent;
11 import md.openmoney.rest.dto.graph.ContractEdgeDTO;
12 import md.openmoney.rest.dto.graph.EdgeDTO;
13 import md.openmoney.rest.dto.graph.VertexDTO;
14 import md.openmoney.rest.service.GraphProcessingService;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.http.HttpStatus;
17 import org.springframework.http.ResponseEntity;
18 import org.springframework.web.bind.annotation.*;
19
20 import com.tinkerpop.blueprints.Vertex;
21 import com.tinkerpop.gremlin.java.GremlinPipeline;
22 import com.tinkerpop.pipes.PipeFunction;
23 import com.tinkerpop.pipes.branch.LoopPipe;
24
25 import io.swagger.annotations.Api;
26 import io.swagger.annotations.ApiOperation;
27
28 @Api(description = "for graph processing")
29 @RestController
30 @RequestMapping(path = "/graph")
31 public class GraphProcessingController {
32     @Autowired
```

```

33     private ElementStorage elementStorage;
34     @Autowired
35     private InstitutionStorage institutionStorage;
36     @Autowired
37     private ContractStorage contractStorage;
38     @Autowired
39     private GraphProcessingService graphProcessingService;
40
41     @ApiOperation(value = "(DONE) get first level graph from rid")
42     @RequestMapping(value = "/{id:\\d+\\:\\d+}/firstLevel", method =
RequestMethod.GET)
43     public ResponseEntity<Object> getFirstLevelGraph(@PathVariable("id") String
entityRid) {
44
45         GraphContent result = elementStorage.getFirstLevelGraphOfEntity(
entityRid);
46
47         return new ResponseEntity<>(result, HttpStatus.OK);
48     }
49
50     @ApiOperation(value = "(DONE) get relational bidirectional graph from rid
entity")
51     @RequestMapping(value = "/{id:\\d+\\:\\d+}/relationalBidirectional", method
= RequestMethod.GET)
52     public ResponseEntity<Object> getRelationalBidirectionalGraph(@PathVariable(
"id") String entityRid,
53                                                                    @RequestParam(
"depth") int depth) {
54
55         if (depth > 8 || depth < 1) return new ResponseEntity<>(HttpStatus.
PRECONDITION_FAILED);
56
57         GraphContent result = elementStorage.
getRelationalBidirectionalGraphOfEntity(entityRid, depth);
58
59         return new ResponseEntity<>(result, HttpStatus.OK);
60     }
61
62     @ApiOperation(value = "(DONE) get connection graph with schema between
personRid and institutionRid, set maxDepth >= 2")
63     @RequestMapping(value = "/connectionWithSchema", method = RequestMethod.GET,
produces = "application/json; charset=utf-8")
64     public ResponseEntity<Object> getConnectionGraphWithSchema(@RequestParam("
personRid") String personRid,
65                                                                    @RequestParam("
institutionRid") String institutionRid,
66                                                                    @RequestParam(
value = "maxDepth", required = false) Integer maxDepth) {
67

```

```

68         if (maxDepth != null && maxDepth <= 1)
69             return new ResponseEntity<>(HttpStatus.PRECONDITION_FAILED);
70
71         String paths = graphProcessingService.getPathsOfConnectionGraph(
            personRid, institutionRid, maxDepth);
72
73         return new ResponseEntity<>(paths, HttpStatus.OK);
74     }
75 }

```

Listing 2.1 – Graph processing controller

- Service Layer: contains any business logic such as validations, defines the scope of business transactions

In Listing 2.2 is service which is responsible for implementing all necessary business logic for exposing shortest paths between some entities with delegating some routine tasks like retrieving data to DAO layer.

```

1 package md.openmoney.rest.service;
2
3 import com.fasterxml.jackson.core.JsonProcessingException;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import md.openmoney.rest.dao.ContractStorage;
6 import md.openmoney.rest.dao.ElementStorage;
7 import md.openmoney.rest.dao.InstitutionStorage;
8 import md.openmoney.rest.dto.graph.ContractEdgeDTO;
9 import md.openmoney.rest.dto.graph.EdgeDTO;
10 import md.openmoney.rest.dto.graph.VertexDTO;
11 import org.springframework.cache.annotation.Cacheable;
12
13 import java.util.*;
14
15 public class GraphProcessingService {
16
17     private ElementStorage elementStorage;
18     private InstitutionStorage institutionStorage;
19     private ContractStorage contractStorage;
20     private ObjectMapper objectMapper = new ObjectMapper();
21
22     public GraphProcessingService(){}
23
24     public GraphProcessingService(ElementStorage elementStorage,
25                                   InstitutionStorage institutionStorage,
26                                   ContractStorage contractStorage) {
27         this.elementStorage = elementStorage;
28         this.institutionStorage = institutionStorage;
29         this.contractStorage = contractStorage;
30     }

```

```

31
32 // @Cacheable(value = "pathsOfConnectionGraph", key="#personRid + #
institutionRid")
33 public String getPathsOfConnectionGraph(String personRid,
34                                         String institutionRid,
35                                         Integer maxDepth) {
36     // check if exists aggregated connection
37     Map<String, Object> aggregatedConnection = elementStorage.
getAgregatedConnection(personRid, institutionRid);
38
39     String pathsResult = null;
40
41     if (aggregatedConnection != null) {
42         pathsResult = aggregatedConnection.get("pathsResult").toString();
43     } else {
44         List<String> companyRidsWhichHaveContractWithInstitution =
elementStorage
45             .getCompanyRidsWhichHaveContractWithInstitution(
institutionRid);
46
47         List<Map<String, Object>> paths = new ArrayList<>();
48
49         companyRidsWhichHaveContractWithInstitution
50             .forEach(companyRid -> {
51             List<VertexDTO> shortestPath = elementStorage.
getShortestPath(
52                 personRid, companyRid.substring(1), maxDepth);
53
54             if (shortestPath == null || shortestPath.size() < 2)
return;
55
56             // start graph
57             Set<VertexDTO> vertices = new HashSet<>();
58             Set<EdgeDTO> edges = new HashSet<>();
59
60             // add last vertex first of all
61             vertices.add(shortestPath.get(shortestPath.size() - 1));
62
63             for (int i = 0; i < shortestPath.size() - 1; i++) {
64                 vertices.add(shortestPath.get(i));
65
66                 String firstRid = shortestPath.get(i).getId().
substring(1);
67                 String secondRid = shortestPath.get(i + 1).getId().
substring(1);
68
69                 List<EdgeDTO> betweenEdges = elementStorage.
getEdgesBetween(firstRid, secondRid);
70                 edges.addAll(betweenEdges);

```

```

71         }
72
73         EdgeDTO contractEdge = new ContractEdgeDTO(
74             companyRid + "-" + institutionRid, "Contract",
75             companyRid, "#" + institutionRid,
76             (int)contractStorage.
getSumOfContractsBetweenCompanyAndInstitution(
77                 companyRid.substring(1), institutionRid
78             )
79         );
80         edges.add(contractEdge);
81
82         Map<String, Object> institutionInfo = institutionStorage
.getInstitutionById(institutionRid);
83         VertexDTO institution = new VertexDTO("#" +
institutionRid, "Institution", institutionInfo.get("name").toString().replace
84             ("\\", "-"));
85
86         vertices.add(institution);
87
88         // end graph
89         Map<String, Object> graph = new HashMap<>();
90         graph.put("vertices", vertices);
91         graph.put("edges", edges);
92
93         // append graph
94         Map<String, Object> path = new HashMap<>();
95         path.put("depth", shortestPath.size());
96         path.put("graph", graph);
97         paths.add(path);
98     });
99     Collections.sort(paths, (p1, p2) -> ((Integer)p1.get("depth")).
compareTo(((Integer)p2.get("depth"))));
100
101     try {
102         pathsResult = objectMapper.writeValueAsString(paths);
103         System.out.println(pathsResult);
104     } catch (JsonProcessingException e) {
105         e.printStackTrace();
106     }
107
108     elementStorage.saveAgregatedConnection(personRid, institutionRid,
pathsResult);
109
110     }
111     return pathsResult;
112 }

```

Listing 2.2– Graph processing Service

- Persistence Layer: maps the database to/from in-memory domain objects

In Listing 2.3 is DAO element which is responsible for interaction with database to retrieve some aggregated data about suspicious cases and exposing those data to service.

```
1 public class RelationEntityDAOImpl {
2
3     private FramedGraph<OrientGraph> graph;
4     private OrientGraphFactory orientGraphFactory;
5     private NamingService namingService;
6
7     public RelationEntityDAOImpl(
8         FramedGraph<OrientGraph> graph,
9         OrientGraphFactory orientGraphFactory,
10        NamingService namingService
11    ) {
12        this.graph = graph;
13        this.orientGraphFactory = orientGraphFactory;
14        this.namingService = namingService;
15    }
16
17    public GraphContent getGraphOfSuspiciousCase(String fromRid, String toRid) {
18        OrientGraph orientGraph = null;
19        try {
20            orientGraph = orientGraphFactory.getTx();
21
22            String query = "select *, eval(\"@class instanceof 'V'\") as isVertex from (
23                select expand($a) +
24                    \" let $a = intersect(\" +
25                    \"     (select from (traverse outE('contractInstitution','companyContract','
26                    companyFounder','personFounder','administrator'), inV() from #\" + fromRid + \"
27                    while true)),\" +
28                    \"     (select from (traverse inE('contractInstitution','companyContract','
29                    companyFounder','personFounder','administrator'), outV() from #\" + toRid + \" while
30                    true))\" +
31                    \" ) )\";
32
33            OCommandSQL commandSQL = new OCommandSQL(query);
34            Iterable<Element> elementsIt = orientGraph.command(commandSQL).execute();
35            List<Element> elements = Lists.newLinkedList(elementsIt);
36
37            List<Map<String, Object>> vertices = elements
38                .stream()
39                .filter(element -> (boolean)element.getProperty(\"isVertex\"))
40                .map(v -> {
41                    Map<String, Object> map = new HashMap<>();
42
43                    map.put(\"id\", v.getId().toString());
44                    map.put(\"type\", v.getProperty(\"@class\"));
```

```

40
41         if (v.getProperty("@class").equals("Contract")) {
42             map.put("sum", v.getProperty("sum"));
43             map.put("name", v.getProperty("objectDescription"));
44         } else {
45             map.put("name", namingService.getMoreClearEntityName((Vertex)v));
46         }
47         return map;
48     })
49     .collect(Collectors.toCollection(LinkedList::new));
50
51     List<Map<String, Object>> edges = elements
52         .stream()
53         .filter(element -> !(boolean)element.getProperty("isVertex"))
54         .map(e -> {
55             Map<String, Object> map = new HashMap<>();
56
57             map.put("id", e.getId().toString());
58             map.put("from", ((OrientVertex) e.getProperty("out")).getId().toString())
59             ;
60             map.put("to", ((OrientVertex) e.getProperty("in")).getId().toString());
61             map.put("type", e.getProperty("@class"));
62             return map;
63         })
64         .collect(Collectors.toCollection(LinkedList::new));
65
66     return new GraphContent(vertices, edges);
67 } finally {
68     if (orientGraph != null) {
69         orientGraph.shutdown();
70     }
71 }
72 }

```

Listing 2.3– Element Storage DAO

Also there is DB layer and in Listing 2.4 is present context configuration for database transaction objects.

```

1 <beans xmlns="http://www.springframework.org/schema/beans"
2     xmlns:context="http://www.springframework.org/schema/context"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mvc="http://www.
4     springframework.org/schema/mvc"
5     xmlns:cache="http://www.springframework.org/schema/cache"
6     xsi:schemaLocation="http://www.springframework.org/schema/beans
7     http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
8     http://www.springframework.org/schema/context
9     http://www.springframework.org/schema/context/spring-context-4.3.xsd http://www.

```



```
springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-
mvc.xsd http://www.springframework.org/schema/cache http://www.springframework.org
/schema/cache/spring-cache.xsd">
```

```

9
10 <bean class="com.tinkerpop.blueprints.impls.orient.OrientGraph">
11     <constructor-arg type="java.lang.String"
12         value="${orientDB.protocol}:${orientDB.host}:${orientDB.port}/${orientDB.name}"
13     />
14 </bean>
15
16 <bean class="com.tinkerpop.frames.FramedGraphFactory">
17     <constructor-arg>
18         <list>
19         </list>
20     </constructor-arg>
21 </bean>
22
23 <bean class="com.tinkerpop.frames.FramedGraph" factory-bean="com.tinkerpop.frames.
24     FramedGraphFactory"
25     factory-method="create">
26     <constructor-arg ref="com.tinkerpop.blueprints.impls.orient.OrientGraph"/>
27 </bean>
28
29 <!--<bean id="orientDbConnectionPool" class="com.orienttechnologies.orient.core.db.
30     OPartitionedDatabasePool">-->
31     <!--<constructor-arg type="java.lang.String" value="${orientDB.protocol}:${
32         orientDB.host}:${orientDB.port}/${orientDB.name}"/>-->
33     <!--<constructor-arg type="java.lang.String" value="${orientDB.user}"/>-->
34     <!--<constructor-arg type="java.lang.String" value="${orientDB.pass}"/>-->
35     <!--<constructor-arg type="int" value="64"/>-->
36     <!--<constructor-arg type="int" value="100"/>-->
37 <!--</bean>-->
38
39 <bean id="orientGraphFactory" class="com.tinkerpop.blueprints.impls.orient.
40     OrientGraphFactory" >
41     <constructor-arg type="java.lang.String" value="${orientDB.protocol}:${orientDB.
42         host}:${orientDB.port}/${orientDB.name}"/>
43     <constructor-arg type="java.lang.String" value="${orientDB.user}"/>
44     <constructor-arg type="java.lang.String" value="${orientDB.pass}"/>
45     <constructor-arg type="boolean" value="true"/>
46     <!--<constructor-arg ref="orientDbConnectionPool"/>-->
47 </bean>
48
49 <!--<bean id="orientGraphR" class="com.tinkerpop.blueprints.impls.orient.
50     OrientGraph"-->
51     <!--scope="request" factory-bean="orientGraphFactory" factory-method="getTx"
52     destroy-method="shutdown"/>-->

```

Listing 2.4– DB context configuration

2.2.2 Client (browser)

The client is MVC-capable and contains all the presentation logic which is separated in a view layer, a controller layer and a frontend services layer. After the initial application startup, only JSON data goes over the wire between client and server.

The frontend should be built around a view-specific model (which is not the domain model), and should only handle presentation logic, but no business logic.

These are the three layers of the frontend:

- View Layer - the view layer is composed of Html templates, CSS, and any Aurelia directives representing the different UI components.

In Listing 2.5 is the html template for rendering of company page.

```

1 <template>
2   <require from="../../../converters/dateConverter"></require>
3   <require from="../../../converters/moneyConverter"></require>
4
5   <compose router.bind="router" view-model="shell/nav-bar"></compose>
6
7   <div class="container">
8
9     <div class="row">
10
11       <compose view-model="shell/page-title" model.bind="{ titleInfo: 'Companii
12         ' }" style="text-align: center"></compose>
13
14     </div>
15
16     <div class="row">
17       <div class="col-xs-12">
18         <div class="search">
19           <div class="col-md-offset-3 col-md-5 col-sm-offset-2 col-sm-8 col-xs-8
20             form-group top_search">
21             <div class="input-group">
22               <form submit.delegate="search()">
23                 <input value.bind="name" type="text" class="form-control"
24                   placeholder="Denumire sau IDNO">
25                 <span class="input-group-btn">
26                   <button class="btn btn-default" type="submit">ăCaut!</

```

```

27         </div>
28     </div>
29 </div>
30 </div>
31
32 <div class="row">
33     <div class="col-md-10 col-md-offset-1 col-xs-12">
34         <div class="x_panel">
35             <div class="x_title">
36                 <h2> <span class="{iconClass}"></span>
37                     {tableTitle}
38                 <small>
39                     {description}
40                 </small>
41             </h2>
42             <div class="clearfix"></div>
43         </div>
44         <div class="x_content">
45             <div class="table-responsive">
46
47                 <table ref="resultsTable" id.bind="title" class="display table
row-border hover marked stripe dataTables_wrapper form-inline dt-bootstrap no
-footer responsive no-wrap " cellpadding="0" width="100%">
48                     <thead show.bind="head">
49
50                         <tr>
51
52                             <th repeat.for="header of headers">
53                                 {header}
54                             </th>
55
56                         </tr>
57                     </thead>
58
59                     <tbody>
60                         <div if.bind="!companies" class="loader">
61                             
62                         </div>
63                         <tr repeat.for="row of companies">
64                             <td>
65                                 <a href="company/{row.newId}">
66                                     {row.identity}
67                                 </a>
68                             </td>
69                             <td>
70                                 <a href="company/{row.newId}">
71                                     {row.name}
72                                 </a>
73                             </td>

```

```

74         <td>
75             ${row.registrationDate | dateFormat}
76         </td>
77         <td>
78             <strong>${row.agregatedInfo.totalSum | moneyFormat}</strong>
79         </td>
80     </tr>
81 </tbody>
82 </table>
83 <ul class="pagination pagination-sm m-t-none m-b-none">
84     <li repeat.for="item of pagination" if.bind="!item.hide" class.
bind="item.isActive ? 'active' : ''" >
85         <a click.delegate="loadPage(item.value)">
86             <i if.bind="item.prev" class="fa fa-chevron-left"></i>
87             <i if.bind="item.next" class="fa fa-chevron-right"></i>
88             ${item.label}
89         </a>
90     </li>
91 </ul>
92 </div>
93 </div>
94 </div>
95
96 </div>
97 </div>
98
99 </div>
100
101 <footer>
102     <compose view-model="shell/footer"></compose>
103 </footer>
104 </template>

```

Listing 2.5– Company HTML template

– Controller and Service layers

The controller layer is made of Angular controllers that glue the data retrieved from the backend and the view together. The controller initializes the view model and defines how the view should react to model changes and vice-versa.

A set of Aurelia services that allow to interact with the backend and that can be injected into Aurelia controllers.

In Listing 2.6 is the service to retrieve information related to company.

```

1 import {inject} from 'aurelia-framework';
2 import {HttpClient, json} from 'aurelia-fetch-client';
3 import {host} from 'host';
4 import {Router} from 'aurelia-router';

```

```

5 import ultimatePagination from 'ultimate-pagination';
6
7 @inject(Element, HttpClient, host, Router)
8
9 export class Companies {
10     companies = null;
11     text = 'simple';
12     name = '';
13     page = 1;
14     page_size = 10;
15     tableTitle = "Top Companii";
16     description = "2015 - 2016";
17     iconClass = "";
18     head = true;
19     headers = ['IDNO', 'Denumirea', 'Data de inregistrare', 'Suma contractelor (
        MDL)'];
20     lastQuery = '';
21
22     constructor(element, http, host, router) {
23         this.element = element;
24         this.http = http;
25         this.host = host;
26         this.router = router;
27
28     }
29
30
31     activate() {
32         var url = this.lastQuery = this.host + '/company/top?by=totalSum&limit=10&
            page=';
33         this.getCompaniesByQuery(url, 1);
34     }
35     search() {
36
37         this.description = 'conform ăăcutrii - ' + this.name;
38
39         this.name = this.name.normalize('NFC');
40
41         if(this.name && this.name.length > 0) {
42             this.companies = null;
43             var url = this.lastQuery = this.host + '/company/search?sorted_by=name&
                limit=' + this.page_size + '&text=' + this.name + '&page=' ;
44             this.getCompaniesByQuery(url, 1);
45             this.tableTitle = 'Rezultate';
46         }
47     }
48
49     getCompaniesByQuery(url, page) {
50         return this.http.fetch(url + page)

```

```

51     .then(resp => resp.json())
52     .then(data => {
53         this.companies = data.content;
54         this.companies.map(el => el.newId = el.id.substring(1));
55         this.currentPage = page;
56         this.totalPages = data.totalPages;
57         if (data.totalPages > 1) {
58             this.drawPagination();
59         } else {
60             this.pagination = null;
61         }
62     }).catch(e => {
63         console.log(e);
64         this.companies = [];
65     });
66 }
67
68 drawPagination() {
69
70     this.pagination = ultimatePagination.getPaginationModel({
71         currentPage: this.currentPage,
72         totalPages: this.totalPages
73     });
74
75     this.pagination.map(item => {
76         switch (item.type) {
77             case 'PAGE':
78                 item.label = item.value;
79                 break;
80             case 'ELLIPSIS':
81                 item.label = '...';
82                 break;
83             case 'FIRST_PAGE_LINK':
84                 item.hide = true;
85                 break;
86             case 'LAST_PAGE_LINK':
87                 item.hide = true;
88                 break;
89             case 'PREVIOS_PAGE_LINK':
90                 item.prev = true;
91                 break;
92             case 'NEXT_PAGE_LINK':
93                 item.next = true;
94                 break;
95         }
96     });
97
98 }
99

```

```

100     loadPage(value) {
101         this.getCompaniesByQuery(this.lastQuery, value);
102     }
103 }

```

Listing 2.6 – Company service to interact with REST

2.2.3 Used technologies

- BACKEND: used technologies:
 - Java - as main language
 - Spring DI framework for wiring and in general for management of application components
 - Spring MVC framework as web layer for request handling
 - OrientDB - nosql graph database
 - IntelliJ IDEA - IDE for java development from JetBrains team
- FRONTEND: used technologies:
 - Javascript - as main language
 - Aurelia - javascript framework for browser application
 - Webstorm - IDE for javascript development from JetBrains team
- JAVA

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers 'write once, run anywhere', meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. The major characteristics of Java are:

The created programs are portable in a network. The source program is compiled into what Java calls bytecode, which can be run anywhere in a network on a server or client that has a Java virtual machine. The Java virtual machine interprets the bytecode into code that will run on the real computer hardware. This means that individual computer platform differences such as instruction lengths can be recognized and accommodated locally just as the program is being executed. Platform-specific versions of the program are no longer needed. The code is robust, here meaning that, unlike programs written in C++ and perhaps some other languages, the Java objects can contain no references to data external to themselves or other known objects. This ensures that an instruction can not contain the address of data storage in another application or in the operating system itself, either of which would cause the program and

perhaps the operating system itself to terminate or "crash." The Java virtual machine makes a number of checks on each object to ensure integrity. Java is object-oriented, which means that, among other characteristics, an object can take advantage of being part of a class of objects and inherit code that is common to the class. Objects are thought of as "nouns" that a user might relate to rather than the traditional procedural "verbs." A method can be thought of as one of the object's capabilities or behaviors. In addition to being executed at the client rather than the server, a Java applet has other characteristics designed to make it run fast. Relative to C++, Java is easier to learn. Java was introduced by Sun Microsystems in 1995 and instantly created a new sense of the interactive possibilities of the Web. Both of the major Web browsers include a Java virtual machine. Almost all major operating system developers (IBM, Microsoft, and others) have added Java compilers as part of their product offerings.

The Java virtual machine includes an optional just-in-time compiler that dynamically compiles bytecode into executable code as an alternative to interpreting one bytecode instruction at a time. In many cases, the dynamic JIT compilation is faster than the virtual machine interpretation.

JavaScript should not be confused with Java. JavaScript, which originated at Netscape, is interpreted at a higher level, is easier to learn than Java, but lacks some of the portability of Java and the speed of bytecode. Because Java applets will run on almost any operating system without requiring recompilation and because Java has no operating system-unique extensions or variations, Java is generally regarded as the most strategic language in which to develop applications for the Web.

– Spring DI

Java components / classes should be as independent as possible of other Java classes. This increases the possibility to reuse these classes and to test them independently of other classes (Unit Testing). To decouple Java components from other Java components the dependency to a certain other class should get injected into them rather than the class itself creates / finds this object. A class A has a dependency to class B if class A uses class B as a variable.

If dependency injection is used then the class B is given to class A via

- the constructor of the class A - this is then called construction injection
- a setter - this is then called setter injection

The general concept between dependency injection is called Inversion of Control. A class should not configure itself but should be configured from outside. A design based on independent classes / components increases the re-usability and possibility to test the software. For example, if a class A expects a Dao (Data Access object) for receiving the data from a database you can easily create another test object which mocks the database connection and inject this object into A to test A without having an actual database connection. A software design based on dependency injection is possible with standard Java. Spring just simplifies the use

of dependency injection by providing a standard way of providing the configuration and by managing the reference to the created objects.

The Spring Framework is a very comprehensive framework. The fundamental functionality provided by the Spring Container is dependency injection. Spring provides a light-weight container, e.g. the Spring core container, for dependency injection (DI). This container lets you inject required objects into other objects. This results in a design in which the Java class are not hard-coupled. The injection in Spring is either done via setter injection or via construction injection. These classes which are managed by Spring must conform to the JavaBean standard. In the context of Spring classes are also referred to as beans or as Spring beans.

The Spring core container:

- handles the configuration, generally based on annotations or on an XML file (XMLBeanFactory)
- manages the selected Java classes via the BeanFactory

The core container uses the so-called bean factory to create new objects. New objects are generally created as Singletons if not specified differently.

In this project Spring DI is used intensively because application has layered architecture with many components. It's very helpful to use this tool for context constructing in outside using these components. It's easy to test each element and gain assurance in quality of application.

– Spring MVC

The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for uploading files. The default handler is based on the @Controller and @RequestMapping annotations, offering a wide range of flexible handling methods. With the introduction of Spring 3.0, the @Controller mechanism also allows to create RESTful Web sites and applications, through the @PathVariable annotation and other features.

"Open for extension..." A key design principle in Spring Web MVC and in Spring in general is the "Open for extension, closed for modification" principle.

Some methods in the core classes of Spring Web MVC are marked final. As a developer you cannot override these methods to supply your own behavior. This has not been done arbitrarily, but specifically with this principle in mind.

In Spring Web MVC is possible to use any object as a command or form-backing object; so there is no need in implementing a framework-specific interface or base class. Spring's data binding is highly flexible: for example, it treats type mismatches as validation errors that can be evaluated by the application, not as system errors. Thus the developer need not duplicate the business objects' properties as simple, untyped strings in the form objects simply to handle

invalid submissions, or to convert the Strings properly. Instead, it is often preferable to bind directly to business objects.

Spring's view resolution is extremely flexible. A Controller is typically responsible for preparing a model Map with data and selecting a view name but it can also write directly to the response stream and complete the request. View name resolution is highly configurable through file extension or Accept header content type negotiation, through bean names, a properties file, or even a custom ViewResolver implementation. The model (the M in MVC) is a Map interface, which allows for the complete abstraction of the view technology. It also gives the possibility to integrate directly with template based rendering technologies such as JSP and Velocity, or directly generate XML, JSON, and many other types of content. The model Map is simply transformed into an appropriate format, such as JSP request attributes, a Velocity template model.

In this project Spring MVC is used as core technology in the module which responds to the clients requests. This technology not only gives the possibility to create an API in less steps but also contains additional modules for working with database. One of the strongest parts that makes Spring one of the leading technologies in WEB industry is its documentation and its community which is raising every day.

– OrientDB

OrientDB is an open source NoSQL database management system written in Java. It is a multi-model database, supporting graph, document, key/value, and object models, but the relationships are managed as in graph databases with direct connections between records. It supports schema-less, schema-full and schema-mixed modes. It has a strong security profiling system based on users and roles and supports querying with Gremlin along with SQL extended for graph traversal. OrientDB uses several indexing mechanisms based on B-tree and Extendible hashing, the last one is known as "hash index", there are plans to implement LSM-tree and Fractal tree index based indexes. Each record has Surrogate key which indicates position of record inside of Array list, links between records are stored either as single value of record's position stored inside of referrer or as B-tree of record positions (so-called record IDs or RIDs) which allows fast traversal (with $O(1)$ complexity) of one-to-many relationships and fast addition/removal of new links. OrientDB is the second most popular graph database according to the DB-Engines graph database ranking.

OrientDB's flexible Multi-Model Database contains a pure Graph Database engine compliant with the Apache TinkerPop standard. OrientDB is incredibly fast (storing up to 120,000 records per second*). It supports schema-less, schema-full and schema-mixed modes and includes SQL among its query languages along with a custom SQL based language which reduces the learning curve for those new to OrientDB. What's more, our graph editor makes creating and editing vertices or edges simple.

In OrientDB all Vertices and Edges are documents. You can embed documents like any other document database, but OrientDB also supports relationships. Why connect documents rather

than embedding them? To avoid duplicates. The resulting database is smaller, lighter and faster, with better use of RAM resulting in more effective caching. Upon loading a tree of documents, OrientDB will assemble the entire document structure by fetching all connections transparently. With OrientDB, traversing speed is not affected by the database size. It is always constant, whether for one record or 100 billion records. This is critical in the age of Big Data!

OrientDB and MongoDB* are both NoSQL databases that share many features, though the engines are fundamentally different. While MongoDB is a pure Document Database, OrientDB is a NoSQL solution with a hybrid Document-Graph engine that adds several compelling features to the Document Database model.

This page will outline only the most important differences.

This is a JSON document representing a simplified Order. Note the “customer” property, which is embedded inside the parent Order object.

OrientDB can embed documents like any other Document Database, but it can also connect documents like a Relational Database. The main difference is that OrientDB doesn’t use the costly JOIN, but rather uses direct, super-fast links taken from the Graph Database world.

The way this transparent fetching of connections is done is one of the strong points of OrientDB. Instead of having repeated calls to the database or costly JOIN operations, a Fetch Plan is given with the query, allowing the database to return a complete graph of interconnected documents, exactly as intended, in a single operation.

MongoDB doesn’t support ACID Transactions. Instead, they support Atomic Operations, so any single operation against documents is atomic. This means that you cannot have Atomicity against multiple documents. For some use cases, this is acceptable, but in others this would be a big problem. OrientDB supports Atomic Operations as well as ACID Transactions, just like the Relational Database model. OrientDB uses a Write Ahead Logging (WAL) Journal to make all the changes durable, even in the event of failure.

MongoDB has its own Query Language based on JSON, which requires training to learn a new language. OrientDB’s query language is built on SQL and is augmented with a few extensions to manipulate trees and graphs. Considering most developers are familiar with SQL, working with OrientDB is easier.

The entire storage of MongoDB is managed using the Memory Mapping technique. This is great, because it’s very fast and managed by the Operating System (OS). In the past, OrientDB used the exact same technique with its “LOCAL” Storage Engine. However, the problem with the Memory Mapping approach is that each OS manages Memory Mapped files in a different way and the available tools to tune the process are very limited and low level. This introduces many problems when databases require more space than the available RAM on the server.

In this project OrientDB is main technology because it support graph model of storage for processing all features.

- IntelliJIDEA

IntelliJ IDEA is a Java integrated development environment for developing computer software. It is developed by JetBrains and is getting more popular with times.

Every aspect of IntelliJ IDEA is specifically designed to maximize developer productivity. Together, powerful static code analysis and ergonomic design make development not only productive but also an enjoyable experience. After IntelliJ IDEA's index source code, it offers blazing fast and intelligent experience by giving relevant suggestions in every context: instant and clever code completion, on-the-fly code analysis and reliable refactoring tools. Smart code completion on the basic completion suggesting names of classes, methods, fields, and keywords within the visibility scope, the smart completion suggests only those types that are expected in the current context. While IntelliJ IDEA is an IDE for Java, it also understands and provides intelligent coding assistance for a large variety of other languages such as SQL, HTML, JavaScript, etc., even when the language expression is injected into a String literal in Java code.

The best part that makes this IDE better than other is about built-in tools and supported frameworks for different needs such as enterprise frameworks, tools for mobile development and web development. Also there is no need in install additional softwares for version control because it automatically have addons for the most popular revision systems such as GIT, Mercurial and SVN.

- Javascript

JavaScript, often abbreviated as "JS", is a high-level, dynamic, untyped, and interpreted run-time language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production; the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

2.3 Database Data Model

The key technology used for this project is "OrientDB" NoSQL DBMS of graph type. The project is about relations between entities and investigation some sort of money flow from institution to other entity into a graph, so this is perfect case to use this DBMS to facilitate its engine of graph processing like shortest path for investigate possible money flow finding shortest relation between a person and institution, traverse hierarchy from an entity to visualize in deep its relations.

Schema is based on classes. OrientDB has two main classes E (Edge) and V (vertex), so you can create subclasses and generate schema of your database, but there is no constraints between

subclasses, so schema of "Openmoney" is represented by a typical case that covers all relations in Figure 2.6, from the left of image is legend of subclasses.

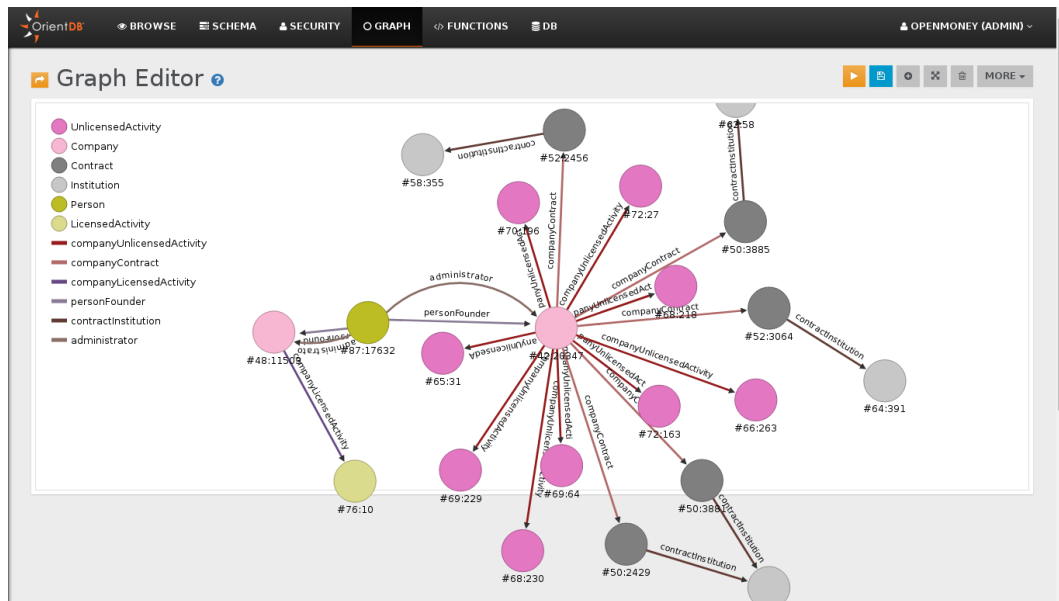


Figure 2.6– Deployment diagram of server environment

2.4 Server Environment

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process. Examples of artifacts are executable files, libraries, archives, database schemas, configuration files, etc. Deployment target is usually represented by a node which is either hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity. In UML artifacts are deployed to nodes, and artifacts could manifest (implement) components. Components are deployed to nodes indirectly through artifacts. Deployment diagrams could describe architecture at specification level (also called type level) or at instance level (similar to class diagrams and object diagrams).

In server environment should be present next services (check Figure 2.7):

- OrientDB (NoSQL graph database)
- Apache Tomcat - servlet container for deployment of REST service component
- Nginx - web server as proxy to REST service for additional stuff of configuration

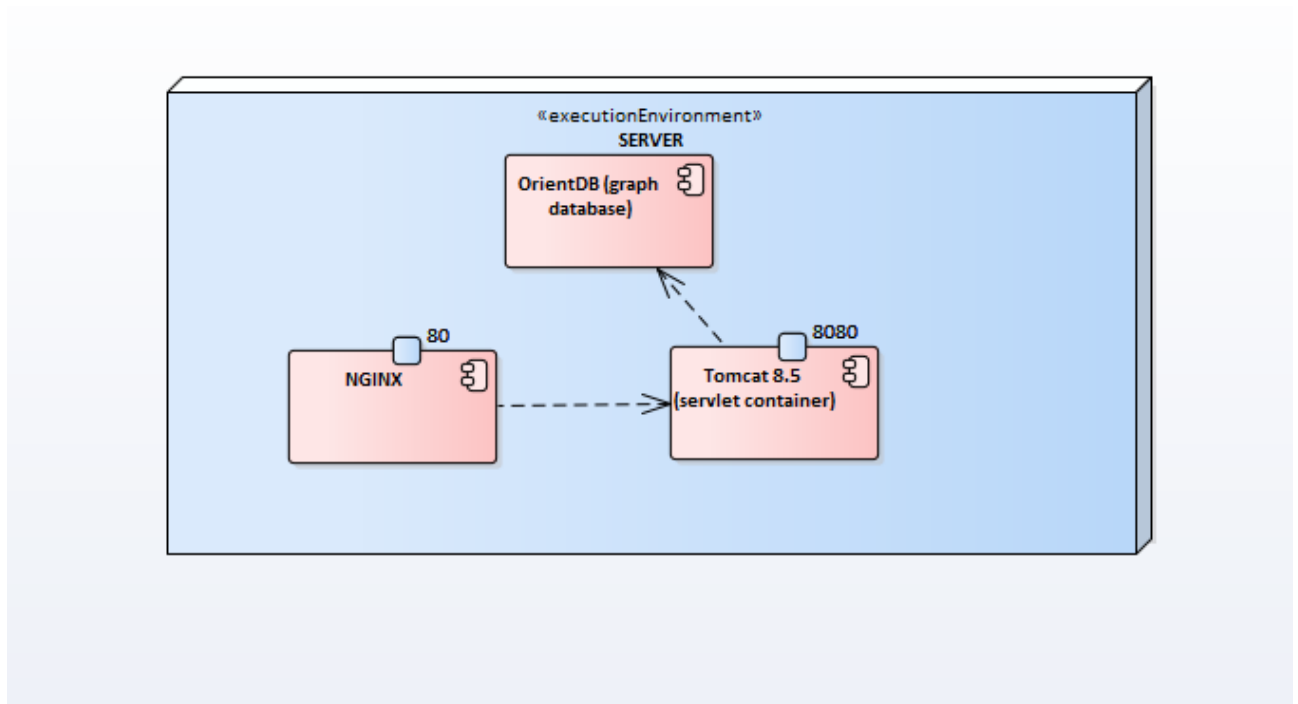


Figure 2.7– Deployment diagram of server environment

3 Review of system in action

In this chapter, in the "Openmoney" application are analyzed already implemented features of project from client point of view.

3.1 Home page

In Figure 3.1 is presented Home Page with next elements:

- Total number of founders, companies and institutions existing for the last time
- Top 10 persons (founders or owners) by sum of contracts of direct associated companies
- Top 10 institutions by sum of contracts
- Top 10 companies by sum of contracts
- Random suspicious case
- Top 10 acquisitions by sum

3.2 Companies

In Figure 3.2 is default page for companies which consists a search bar where you can find a company by name or ID number of that entity and a list of top companies which stay by default. This list contains next columns: IDNO (ID of company), name, date of registration and sum of all acquisitions made by this company for period 2015-2016. Under this list you can see pagination for

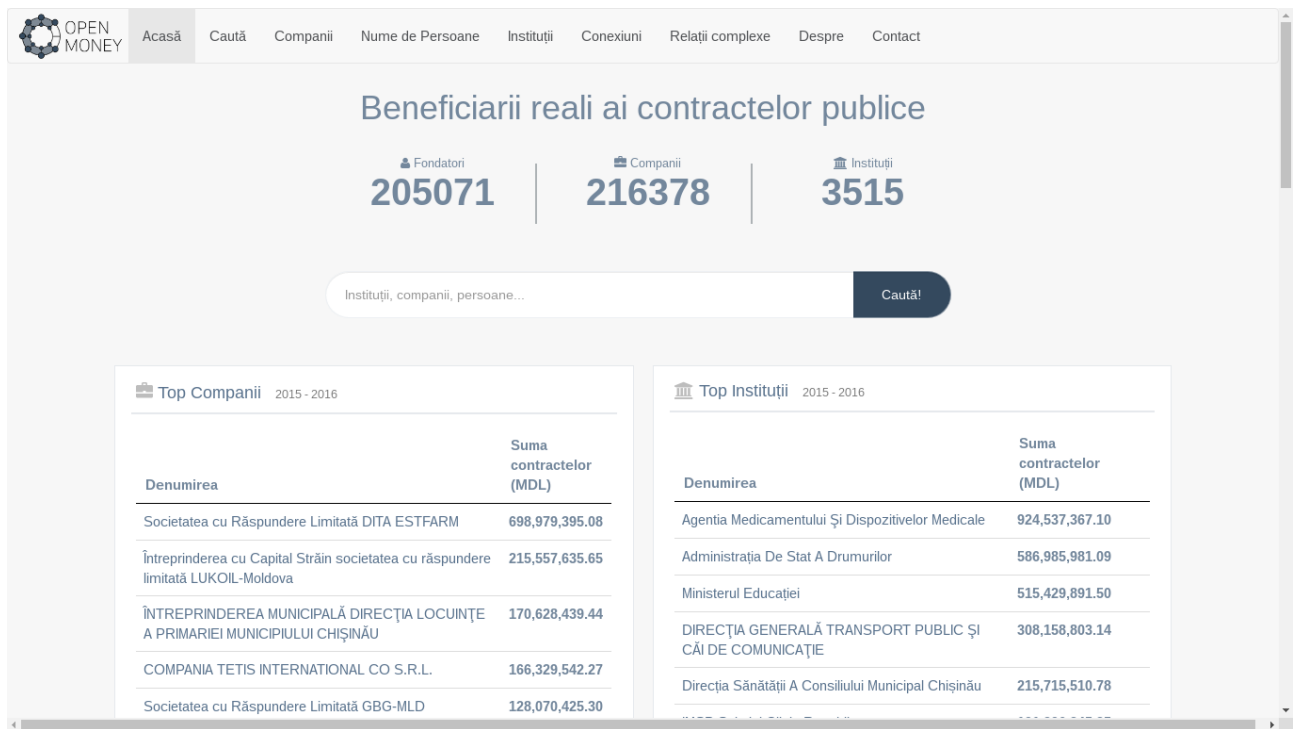


Figure 3.1 – Home Page

all companies which has at least one acquisition with state sorted by sum of money.

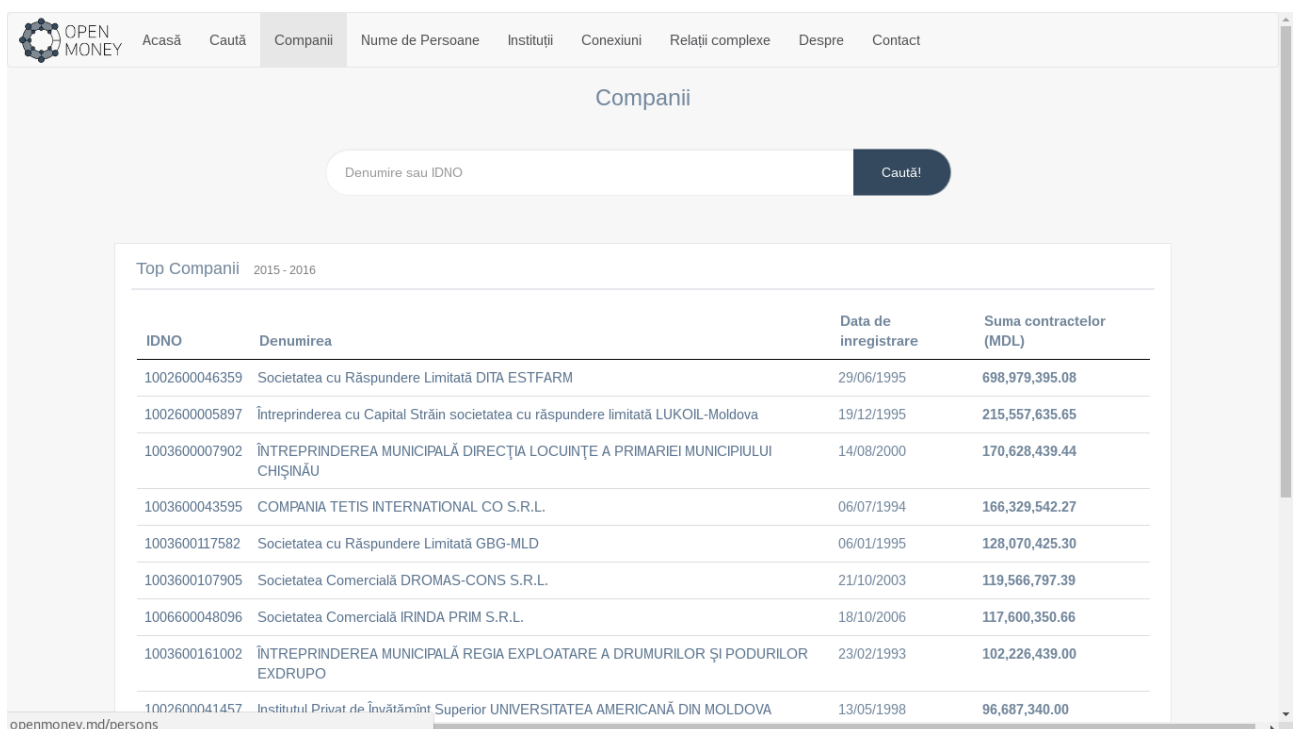


Figure 3.2 – Companies

After clicking on necessary company user goes to another page (Figure 3.3) specific for describing that entity. This page starts from title line with name of company, after that are displaying 3 blocks with main information about company:

- General information - IDNO, registration date, address, state (active/inactive)
- Organisational structure - lists with administrators and founders of that company with hyper-link to open new page with that entity
- Activities - lists of licensed and unlicensed activities that does this company

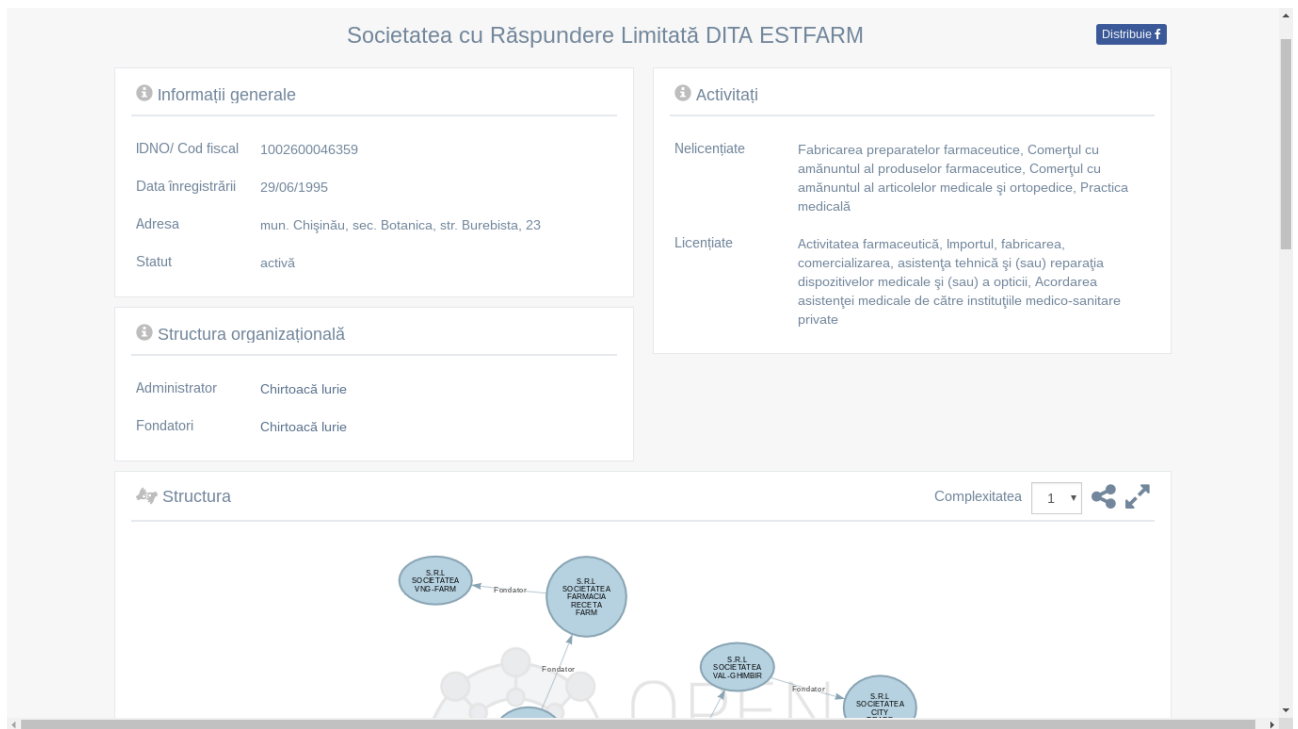


Figure 3.3– Company’s main information

Next block in Figure 3.4 is visual representation of structure(hierarchy of relationship from this entity) in form of graph formed after traversing only in ONE DIRECTION (IN and OUT). There is possibility to scroll to resize canvas, to move elements, to click on entity to do check basic info and do some additional stuff. Also this block has option to change complexity(to traverse graph BIDIRECTIONAL), share as embed this graph, to switch to full screen.

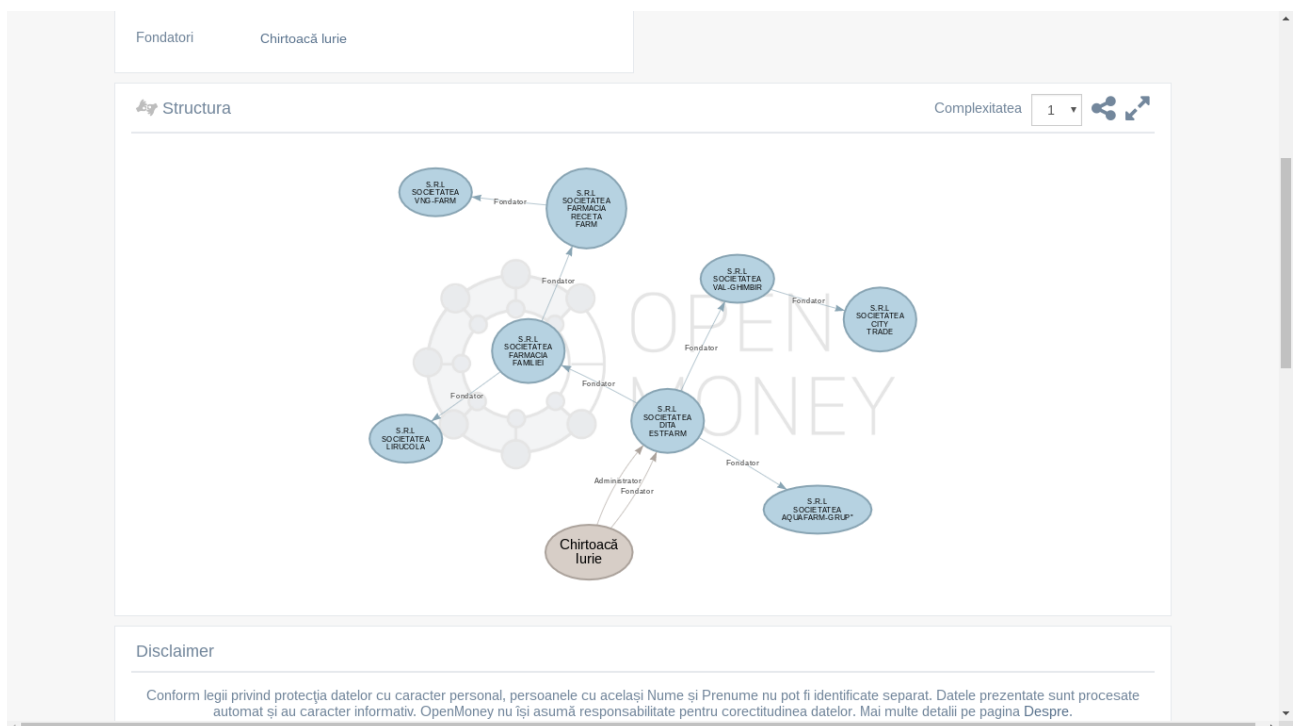


Figure 3.4– Company’s relationship structure traversing only in ONE DIRECTION

In Figure 3.5 is the last block from company page. It represents all acquisitions made by this company with pagination sorted by amount of money spent. In table are such columns as name of institutions that closed acquisitions with this company, date, type, description for exactly what was spent such amount of money and the sum.

Contracte 2015-2016						
Agentia Medicamentului și Dispozitivelor Medicale	DITA ESTFARM S.R.L.	0	Invalid date	De bază	achiziționarea medicamentelor necesare instituțiilor medico sanitare publice (IMSP) pentru anul 2016	295,669,677.40
Agentia Medicamentului și Dispozitivelor Medicale	DITA ESTFARM S.R.L.	7	23/11/2015	De bază	pentru achiziționarea dispozitivelor medicale (articolelor parafarmaceutice) conform necesităților instituțiilor medico-sanitare publice (IMSP) pentru anul 2016	60,262,199.34
IMSP Institutul Oncologic	DITA ESTFARM SRL	0	29/01/2015	Contract de achiziție	medicamente	37,219,993.13
Agentia Medicamentului și Dispozitivelor Medicale	DITA ESTFARM S.R.L.	5	20/05/2015	De bază	pentru achiziționarea articolelor parafarmaceutice conform necesităților instituțiilor medico-sanitare publice (IMSP) pentru anul 2015	20,532,892.09
IMSP Spitalul Clinic Republican	Dita Estfarm SRL	0	29/01/2015	Contract de achiziție	medicamente	13,323,124.08
Agentia Medicamentului și Dispozitivelor Medicale	DITA ESTFARM S.R.L.	0	Invalid date	De bază	○Achiziționarea medicamentelor imunosupresive în scopul realizării Programului Național de Transplant pentru anul 2016	12,995,164.48
Agentia Medicamentului și Dispozitivelor Medicale	Dita Estfarm S.R.L.	0	Invalid date	De bază	achiziționarea medicamentelor necesare instituțiilor medico sanitare publice (IMSP) pentru anul 2016 repetat	11,073,746.96
Agentia Medicamentului și Dispozitivelor Medicale	DITA ESTFARM S.R.L.	0	Invalid date	De bază	Achiziționarea medicamentelor în scopul realizării Programului Național de prevenire, profilaxia și controlul infecției HIV/SIDA/ITS pentru anul 2016	10,144,027.53
IMSP SCM Balti/Balti	DITA	0	11/02/2015	Contract	medicamente	10,044,664.27

Figure 3.5 – Company’s tenders

3.3 Persons (founders and owners)

In Figure 3.6 is default page for persons which consists a search bar where you can find a person by name and a list of top persons which stay by default. This list contains next columns: pseudoID, name and sum of all acquisitions made by his companies for period 2015-2016. Under this list you can see pagination for all persons which has at least one acquisition with state sorted by sum of money.

<div> <div> <div>Acasă</div> <div>Caută</div> <div>Companii</div> <div>Nume de Persoane</div> <div>Instituții</div> <div>Conexiuni</div> <div>Relații complexe</div> <div>Despre</div> <div>Contact</div> </div> </div>																																			
<div>Nume de Persoane</div> <div> <div>Nume Prenume</div> <div>Caută!</div> </div>																																			
<div>Top Nume de Persoane 2015 - 2016</div> <table> <tr> <th>ID</th><th>Nume Prenume</th><th>Suma contractelor (MDL)</th></tr> <tr> <td>#84:3208</td><td>Chirtoacă Iurie</td><td>699,107,795.08</td></tr> <tr> <td>#83:20530</td><td>Isayev Feyruz</td><td>215,557,635.65</td></tr> <tr> <td>#86:5783</td><td>Lukoil Europe Holdings B.v.</td><td>215,557,635.65</td></tr> <tr> <td>#81:23519</td><td>Furtună Mihail</td><td>170,628,439.44</td></tr> <tr> <td>#83:18469</td><td>Mislițchi Svetlana</td><td>170,628,439.44</td></tr> <tr> <td>#88:16072</td><td>Meriacre Vasile</td><td>170,628,439.44</td></tr> <tr> <td>#82:19125</td><td>Ciobu Nadejda</td><td>170,628,439.44</td></tr> <tr> <td>#85:7134</td><td>Cotorobai Nadejda</td><td>170,628,439.44</td></tr> <tr> <td>#81:15265</td><td>Melniciuc Oleg</td><td>170,628,439.44</td></tr> <tr> <td>#85:10548</td><td>Cernei Oleg</td><td>170,628,439.44</td></tr> </table>			ID	Nume Prenume	Suma contractelor (MDL)	#84:3208	Chirtoacă Iurie	699,107,795.08	#83:20530	Isayev Feyruz	215,557,635.65	#86:5783	Lukoil Europe Holdings B.v.	215,557,635.65	#81:23519	Furtună Mihail	170,628,439.44	#83:18469	Mislițchi Svetlana	170,628,439.44	#88:16072	Meriacre Vasile	170,628,439.44	#82:19125	Ciobu Nadejda	170,628,439.44	#85:7134	Cotorobai Nadejda	170,628,439.44	#81:15265	Melniciuc Oleg	170,628,439.44	#85:10548	Cernei Oleg	170,628,439.44
ID	Nume Prenume	Suma contractelor (MDL)																																	
#84:3208	Chirtoacă Iurie	699,107,795.08																																	
#83:20530	Isayev Feyruz	215,557,635.65																																	
#86:5783	Lukoil Europe Holdings B.v.	215,557,635.65																																	
#81:23519	Furtună Mihail	170,628,439.44																																	
#83:18469	Mislițchi Svetlana	170,628,439.44																																	
#88:16072	Meriacre Vasile	170,628,439.44																																	
#82:19125	Ciobu Nadejda	170,628,439.44																																	
#85:7134	Cotorobai Nadejda	170,628,439.44																																	
#81:15265	Melniciuc Oleg	170,628,439.44																																	
#85:10548	Cernei Oleg	170,628,439.44																																	

Figure 3.6 – Persons

In Figure 3.7 is visual representation of structure(hierarchy of relationship from this entity) in form of graph formed after traversing BIDIRECTIONAL in full screen mode.

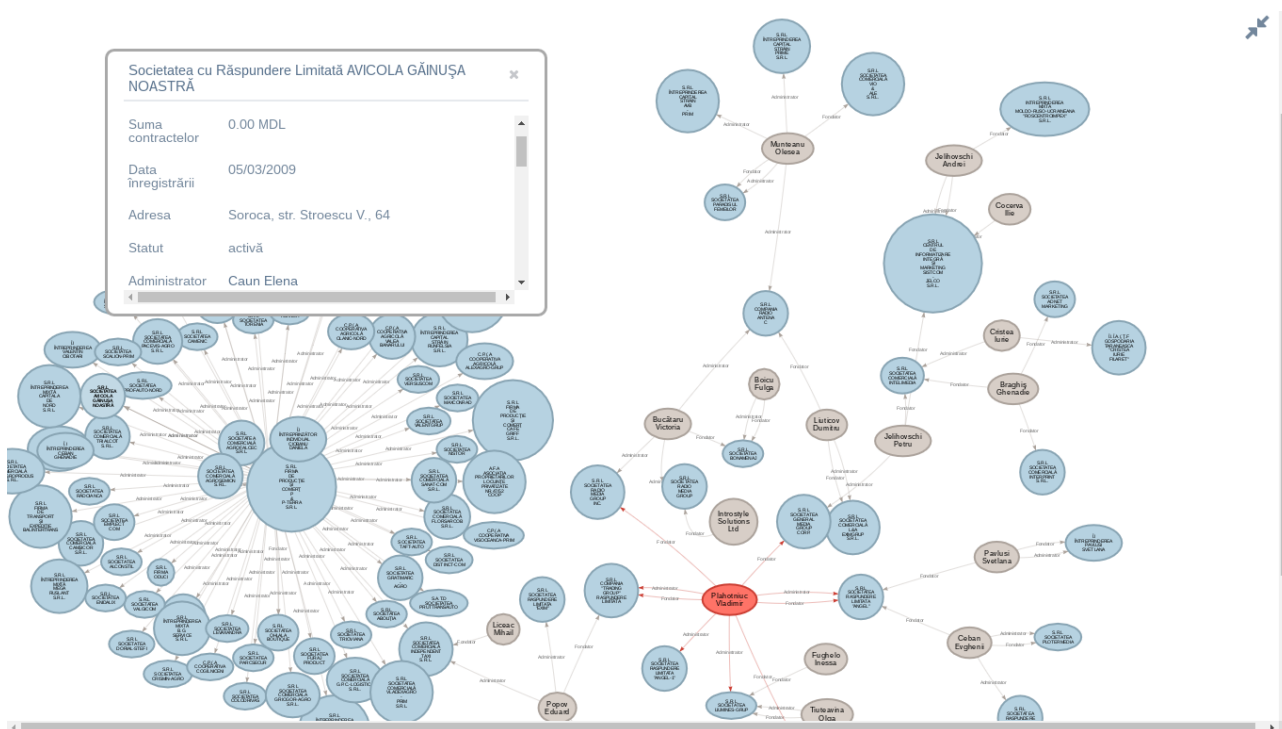
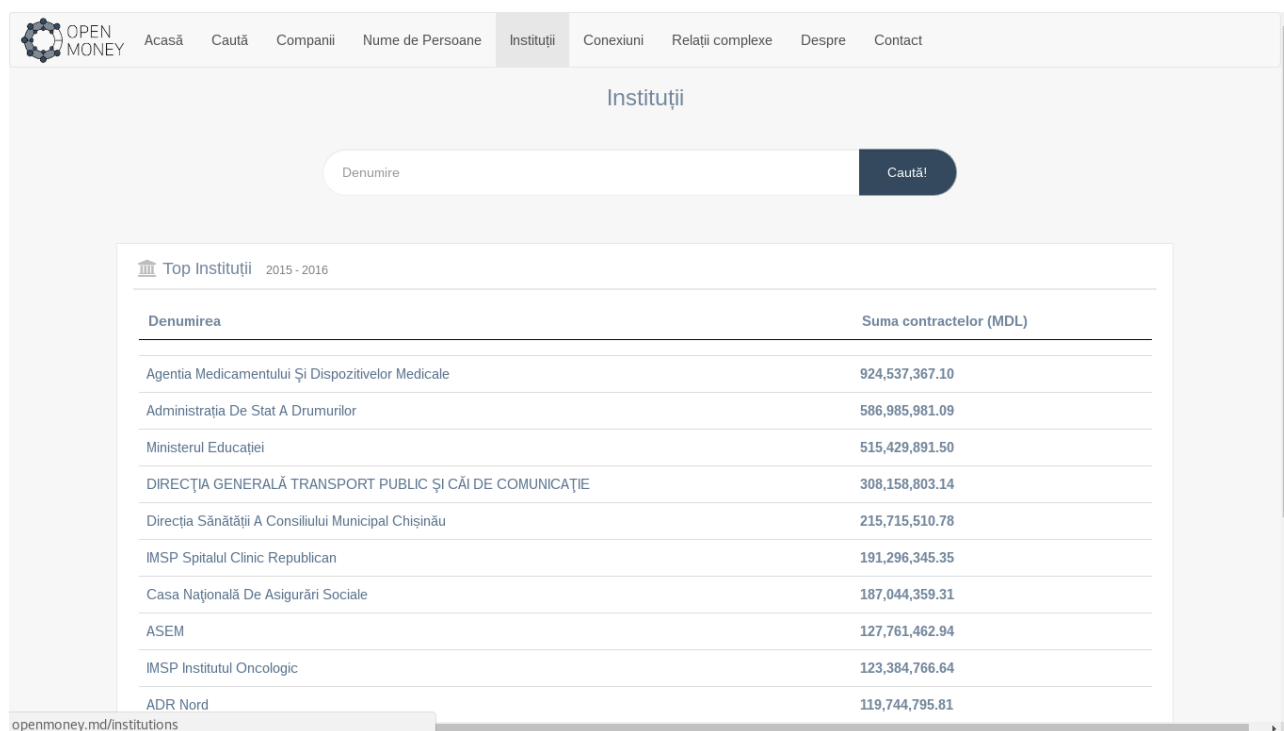


Figure 3.7– Person’s relationship structure with BIDIRECTIONAL traversing

3.4 Institutions

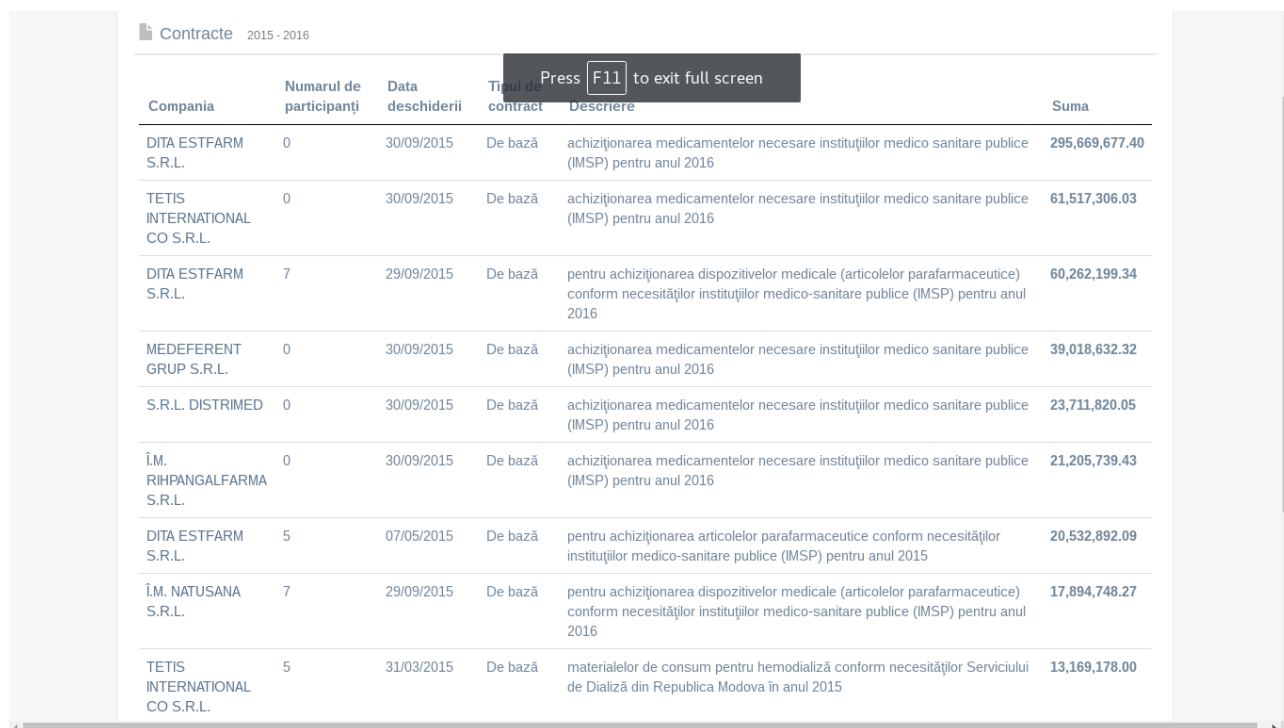
In Figure 3.8 is default page for institutions which consists a search bar where you can find a institution by name and a list of top institutions which stay by default.



Denumirea	Suma contractelor (MDL)
Agentia Medicamentului Și Dispozitivelor Medicale	924,537,367.10
Administrația De Stat A Drumurilor	586,985,981.09
Ministerul Educației	515,429,891.50
DIRECȚIA GENERALĂ TRANSPORT PUBLIC ȘI CĂI DE COMUNICAȚIE	308,158,803.14
Direcția Sănătății A Consiliului Municipal Chișinău	215,715,510.78
IMSP Spitalul Clinic Republican	191,296,345.35
Casa Națională De Asigurări Sociale	187,044,359.31
ASEM	127,761,462.94
IMSP Institutul Oncologic	123,384,766.64
ADR Nord	119,744,795.81

Figure 3.8– Institutions

Acquisitions block in Figure 3.9. with company name and sum.



Compania	Numarul de participant	Data deschiderii	Tipul de contract	Suma
DITA ESTFARM S.R.L.	0	30/09/2015	De bază	295,669,677.40
TETIS INTERNATIONAL CO S.R.L.	0	30/09/2015	De bază	61,517,306.03
DITA ESTFARM S.R.L.	7	29/09/2015	De bază	60,262,199.34
MEDEFERENT GRUP S.R.L.	0	30/09/2015	De bază	39,018,632.32
S.R.L. DISTRIMED	0	30/09/2015	De bază	23,711,820.05
Î.M. RIHPANGALFARMA S.R.L.	0	30/09/2015	De bază	21,205,739.43
DITA ESTFARM S.R.L.	5	07/05/2015	De bază	20,532,892.09
Î.M. NATUSANA S.R.L.	7	29/09/2015	De bază	17,894,748.27
TETIS INTERNATIONAL CO S.R.L.	5	31/03/2015	De bază	13,169,178.00

Figure 3.9– Acquisitions of an institution

3.5 Connections between person and institution

In Figure 3.10 is shown "Connections" module. The idea of it is to choose a person and a institution and to try to find relations if they exist. These relations are shortest paths between companies of the person and institution through BIDIRECTIONAL relations.

Purpose of this features is to find possible money flow from an institution to a person through multiple relations that are hard to find by yourself, but is easier programmatic. Numbers under the button are size of path in sorted order.

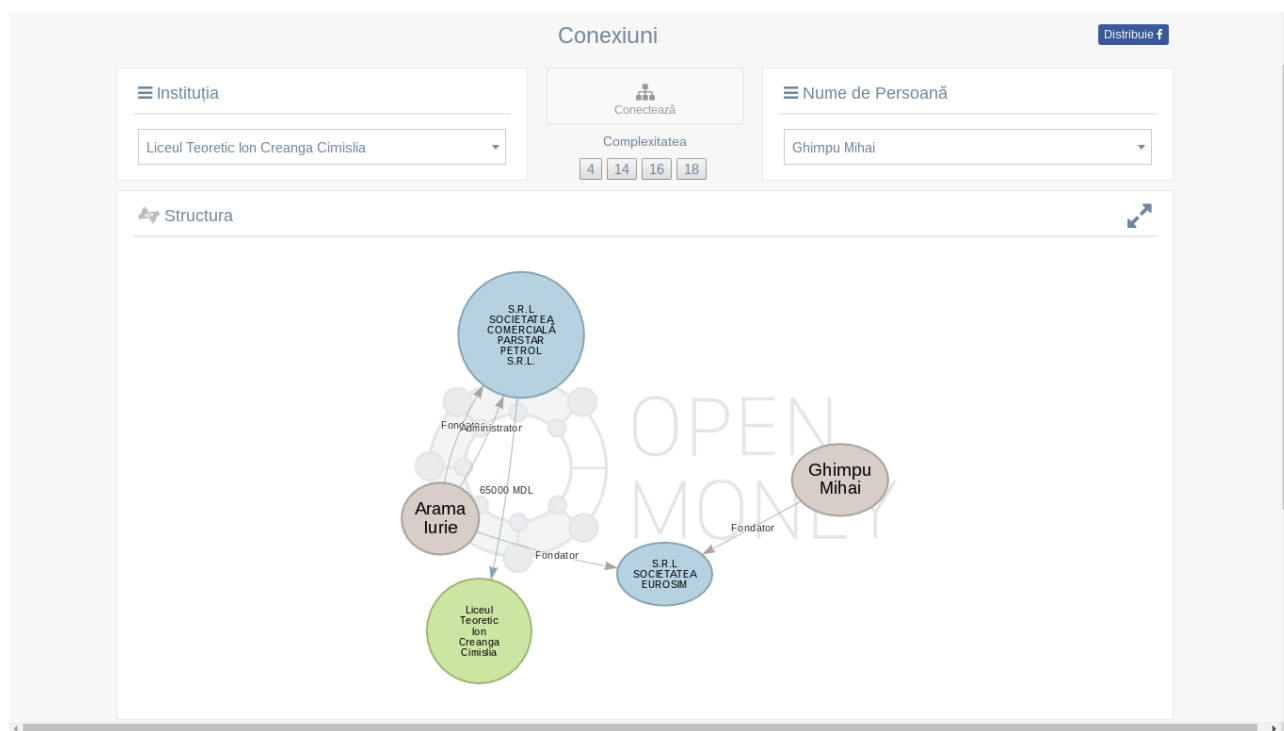


Figure 3.10 – Connections

3.6 Complex relation

Complex relation - automatic detection of relation between person and institution where more companies that won tenders at the same institution are owned by same person in top level hierarchy only through DIRECTED relations, check Figure 3.11.

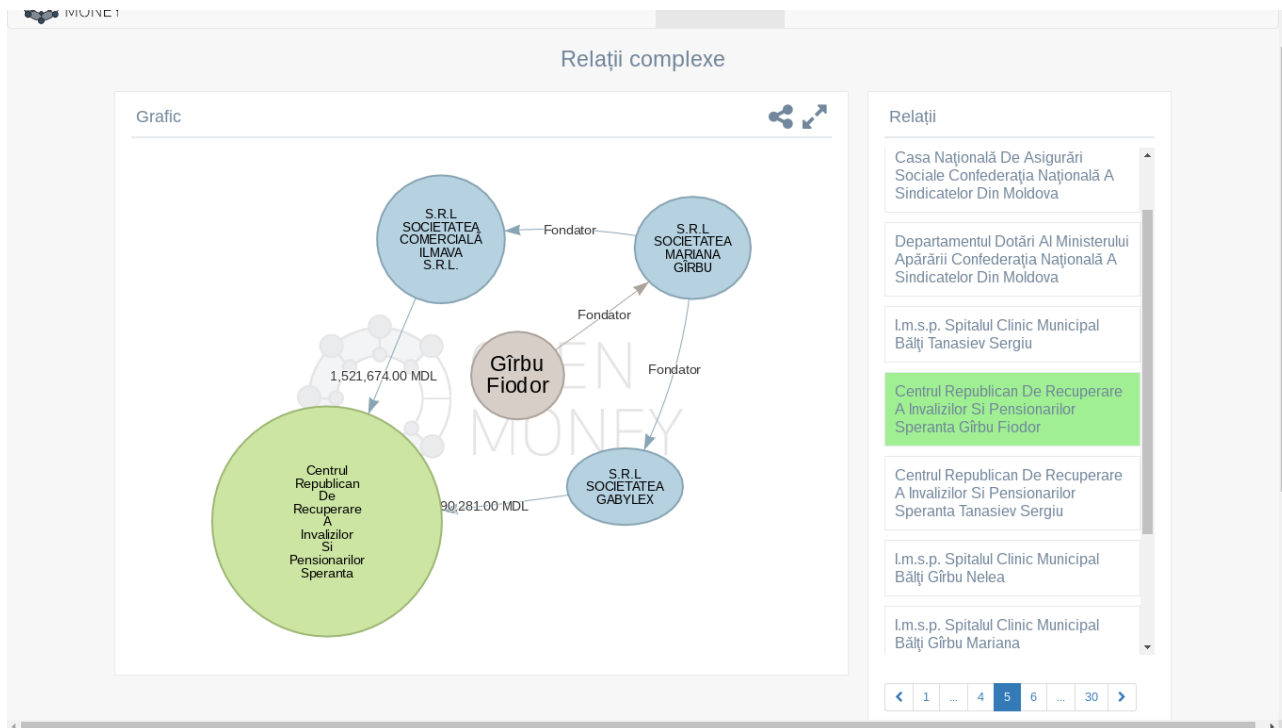


Figure 3.11 – Complex relation

3.7 About

In Figure 3.12 is a page that describes the project and show a tutorial how to use system in youtube video.

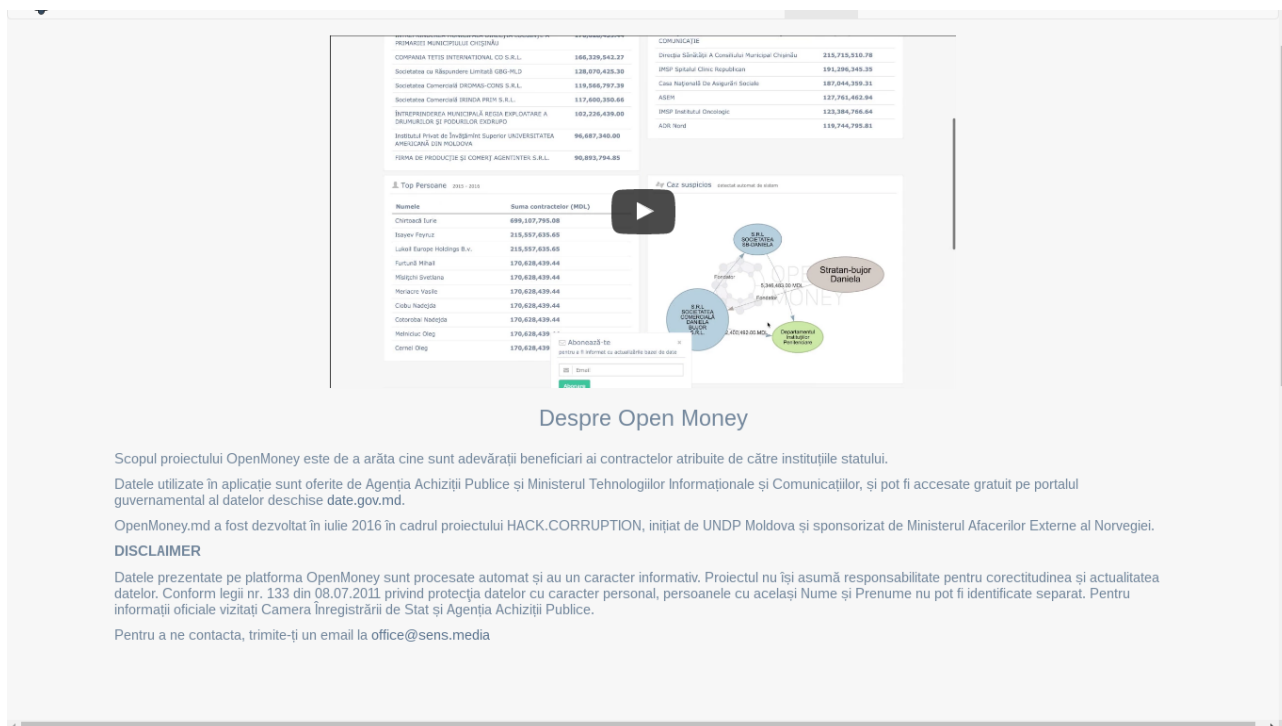


Figure 3.12 – Page about project

4 Economic Evaluation of the Project

4.1 Project description

"Openmoney" is a project for visualizing information about companies and their acquisitions with institutions in a more easier way for simple users and professional journalists.

Development is based on money obtained from grant. After finishing first version need to analize the possible additional features and to try to search for other investments in dependence of wanted features.

Economic motivation The economic motivation of the project is to tend to complicated scheme analysing to find real suspicious cases, then would be possible to share this project for National anticorruption center as a service that could help them.

4.2 SWOT Analysis

SWOT analysis or SWOT matrix is a structured planning method that evaluates Strengths, Weaknesses, Opportunities, and Threats of a project, organization or venture. It can be applied for a wide set of topics such as a company, product, place and so on. The purpose of SWOT analysis is to investigate the internal and external factors that are favorable or unfavorable to achieve the objective. Finding respective SWOTs can be quite important, since it can reveal some later steps that are required for implementation of objective, yet only when the project has been deemed as attainable one, if not it is recommended to switch the objective that is analyzed. Yet, in order to evaluate the objective in each perspective, users of SWOT must ask and answer meaningful question that in turn generate useful information.

From the SWOT analysis it is clear what strengths should application have, and currently what weaknesses should be avoided if possible. Using this analysis it is possible to foresee and plan the development accordingly, to reduce weaknesses and increase their strengths. It is important as well to monitor external changes of factors, that influence the development of product. Mostly the opportunities and threats from SWOT analysis should be used to monitor, using KPI or OCR.

There are five main steps to be accomplished in order to reach a final result for the project in discussion:

- Planning (form some ideas about the design of the project, build some sketches and diagrams, determine amount of needed resources)
- Researching (analyze the market, the target group for which the project is intended, learn technologies for further use)
- Developing (create the actual system, adjust requirements to the used technologies and frameworks)
- Testing (perform debugging in order to ensure the well behavior of the system and integrity of functionality)

Table 4.1 – Analysis of project using SWOT matrix

Project	Strength	Weakness
Openmoney	<ul style="list-style-type: none"> – Simpler visualization of acquisitions between companies and institutions – Simpler visualization foundation and ownership hierarchy of companies – Public access for everyone 	<ul style="list-style-type: none"> – Possible errors in connections after processing – Bad structure of open data for a programmer – Special features like "connections" or "complex relation" may be misunderstood
Project	Opportunities	Threats
Openmoney	<ul style="list-style-type: none"> – First platform of this type in Moldova – Growing interest of open data processing topics for investments – Interested institutions in additional features 	<ul style="list-style-type: none"> – Not required – No insurance in quality of data because of errors

- Deployment (make the application available for users)

The key to a well-scheduled plan, that would lead to the in-time accomplishment of the project, is the proper subdivision of the workload, according to the given resources. Meaning that, every step enumerated above, should be scheduled for a finite period of time, judging by the level of difficulty implied. The planning period should be considered a flexible one, since at this moment some general ideas on the project have to be discussed. Changes are allowed at this step, since requirements are still established. For the researching period, it is important to be open to new ideas for consideration, in order to have a better outcome. The development period should be the most accurate one and should be divided in sprints, for better monitoring of the workflow. In the table bellow are represented all the general steps involving the development of the project. The following adnotations were used: PM – project manager, iD – developer, BD - Backend Developer.

Table 4.2 – Time schedule

Nr	Activity Name	Duration (days)	People involved
1	Project idea definition	3	PM, iD
2	Perform market analysis	2	PM, iD
3	Establish functional features	14	PM, iD
4	Elaborate Use Case Diagrams	5	PM, iD
5	Implement database design	7	PM, BD
6	Implement application design	7	iD
7	Create back-end functionality	28	BD
8	Test back-end functionality	4	BD
9	Develop Application	210	iD
10	Test entire project	7	PM, BD, iD
11	Prepare projects inside application	3	PM, BD, iD
12	Write documentation	7	iD
13	Upload the application to AppStore	2	iD
14	Commercialize the product	3	PM, iD
15	Total time to finish the system	302	

As a generalization, Table 4.2 represents a sketch for the first iteration of the project. Each activity was evaluated with a time stamp and responsible employees were assigned. For the completion of the purposed system including 14 activities was estimated a total amount of time including 120 working days. Bellow are indicated the amount of spent time for each individual:

- PM: 44 days;
- BD: 49 days;
- iD: 263 days.

4.3 Economic motivation

It is actual in economy to bring economical proofs for the IT projects, basing on the specific of the concurrences in economic relationships, which suppose a wide research space. In the conditions of a low degree of determination of the marketing environment, of high prices' volatility, decreased degree of prognoses depth, a common business-plan doesn't allow the exact foreseeing of the final results of the business. In this context, one of the basic instruments is choosing the methods, the right positions and index for the economical proofs. Realization of this goal conditions a large number of scientifically research, subordinated to the primary goal and formulated by means of the following objectives:

- Studying the theoretical and methodical aspects of the business-planning in the conditions of the concurrency on the market;

- Systematization, determining the methodology and specifying the index for the economical proof of the business-plans in IT;
- Study and analysis of the actual practice of economical proofs of the business-plans for IT in Republic of Moldova;
- Developing methodological concepts of the proofs of the decision of investment in the conditions of risk and incertitude;
- Studying the evaluation criteria of the business-projects' efficiency and elaboration of a mechanism of complex evaluation of these.

4.3.1 Tangible and intangible asset expenses

Expenses and initial budget are the ones that sharpen from the beginning the project itself, by imposing some limitations on the complexity of the system and defining its boundaries. In this section, an evaluation of the necessary amount of money will be computed, in order to ensure the correct administration of financial resources. At the beginning, in Table 4.3, will be listed all the tangible assets used for the project. Tangible assets are defined as any assets that have a physical form.

Table 4.3– Tangible assets expenses

Material	Specification	Measurement unit	Price per unit (MDL)	Quantity	Sum (MDL)
Notebook	MacBook Pro i5 2015	Unit	24000	1	24000
Testing Device	iPhone 5C 16Gb	Unit	5000	1	5000
Cable	Original Lightning Cable	Unit	300	2	600
				Total	29600

The Table 4.4 presents the intangible assets, used for the project. Intangible assets are all those that do not possess a physical form. In this case, the discussion is about the **software** needed to build the application. For development was used *Xcode* IDE, which requires an Apple Developer Account for deploying. For the design of UML diagrams, *Enterprise Architect* was used.

Table 4.4– Intangible asset expenses

Material	Specification	Measurement unit	Price per unit (MDL)	Quantity	Sum (MDL)
License	Apple Developer Program	Unit	2000	2	4000
License	Enterprise Architect Desktop Edition License	Unit	1800	3	5400
				Total	9400

Also, additional expenses represent the direct expenses, used for work facilitation during project development. These costs cannot be included in any of the previous tables, because their values aren't included directly into the budget of the project and they have to be mentioned out of the topic. The Table 4.5 emphasizes the direct expenses.

Table 4.5– Direct expenses

Material	Specification	Measurement unit	Price per unit (MDL)	Quantity	Sum (MDL)
Whiteboard	Universal Dry Erase Board	Unit	700	1	700
Paper	A4	100 sheets	80	1	80
Pen	Blue pen	Unit	5	10	50
				Total	830

Given all the raw data, a total amount of direct expenses can be calculated:

$$T_e = 29600 + 9400 + 830 = 39830 \quad (4.1)$$

4.3.2 Salary expenses

In this compartment will be discussed the remuneration of each employee during the development of the project. It is assumed that each member of the team has the salary listed below:

- Project Manager – 300 MDL
- Backend Developer – 300 MDL
- iOS Developer – 400 MDL

After the closing of the project, some statistics regarding financial costs for employee remuneration can be presented in Table 4.6.

Table 4.6– Salary expenses

Employee	Working days	Salary per day (MDL)	Salary fund (MDL)
Project Manager	44	300	13200
Backend Developer	49	300	14700
iOS Developer	263	400	105200
Total			133100

Besides salaries, the social service fund also retrieves a part of the money, constituting 23% of total salary. Also, there is the medical insurance fund, constituting 4,5% of the same sum. The next step is to compute the social service fund, according to the relation (4.2) :

$$\begin{aligned}
 FS &= F_{re} \cdot T_{fs} \\
 &= 133100 \cdot 0.23 \\
 &= 30613,
 \end{aligned} \quad (4.2)$$

where FS is the salary expense, F_{re} is the salary expense fund and T_{fs} is the social service tax

approved each year. The medical insurance fund is computed as:

$$\begin{aligned}
 MI &= F_{re} \cdot T_{mi} \\
 &= 133100 \cdot 0.045 \\
 &= 5989.5,
 \end{aligned} \tag{4.3}$$

where T_{mi} is the mandatory medical insurance tax approved each year by law of medical insurance.

Now, the total work expense fund is calculated as sum of the previous computed indicators:

$$\begin{aligned}
 WEF &= F_{re} + FS + MI \\
 &= 133100 + 30613 + 5989.5 \\
 &= 169702.5,
 \end{aligned} \tag{4.4}$$

where WEF is the work expense fund, FS is the social fund and MI is the medical insurance fund. The final indicator shows the total work expense fund.

4.4 Individual person salary

Having the total work expense fund computed, it is necessary to determine the net salary for the developer. Considering the developer's salary of 400 MDL per day and there is a totally 120 working days for accomplishing the project, so the gross salary that the developer gets is:

$$GS = 400 \cdot 263 = 105200, \tag{4.5}$$

where GS is the gross salary computed in MDL.

Social fund tax this year represents 6%, so the amount that should be tax paid in MDL represents

$$SF = 105200 \cdot 0.06 = 6312. \tag{4.6}$$

Medical insurance tax represents 4.5% and gives the following result

$$MIF = 105200 \cdot 0.045 = 4734. \tag{4.7}$$

In order to proceed with income tax computations, it is necessary to calculate the amount of taxed salary.

$$\begin{aligned}
 TS &= GS - SF - MIF - PE \\
 &= 105200 - 6312 - 4734 - 10128 \\
 &= 84026,
 \end{aligned} \tag{4.8}$$

where TS is the taxed salary, GS – gross salary, SF – social fund, PE – personal exemption, which this year is approved to be 10128.

The last but not the least thing to be computed is the total income tax, which is 7% for income

under 31140 MDL and 18% for income over 31140 MDL.

$$\begin{aligned}
 IT &= TS - ST \\
 &= 31140 \cdot 0.07 + (84026 - 31140) \cdot 0.18 \\
 &= 2179.8 + 9519.5 = 11699.3,
 \end{aligned} \tag{4.9}$$

where IT is the income tax, TS – the taxed salary and ST – the salary tax.

With all this now it is possible to find out what's going to be the net income.

$$\begin{aligned}
 NS &= GS - IT - SF - MIF \\
 &= 105200 - 11699.5 - 6312 - 4734 \\
 &= 82454.7,
 \end{aligned} \tag{4.10}$$

where NS is the net salary, GS – gross salary, IT – income tax, SF – social fund, MIF – medical insurance fund.

4.4.1 Indirect expenses

Other expenses involved in the completion of the project involves production consumption. In the Table 4.7 are mentioned expenses like public transport, electricity, access and office water.

Table 4.7– Indirect expenses

Material	Specification	Measurement unit	Price per unit (MDL)	Quantity	Sum (MDL)
Internet	Moldtelecom	Pack	200.00	4	800
Transport	Public maxi-taxi	Trip	3.00	240	720
Electricity	Union Fenosa	KWh	1.99	500	502
Total					2022

4.4.2 Wear and depreciation

When speaking about economic analysis, it is vital to understand the influence of time over the product. Usually, it happens that a depreciation in value can appear, that's why it is a value that should be computed and take into account. As a tuple, with the depreciation, the wear will be calculated. Depression will be computed uniformly for the whole project duration, so that there are no accountancy issues. As a matter of fact, a business plan divided for 3 years should be compartmentalized into 3 uniform parts according to each year.

Normally wear is computed regarding to the type of asset. The computer mentioned above as tangible asset can be used for a period of 3 years. Licenses will last for a single year. Straight line depreciation will be applied. First step is to sum up tangible and intangible assets, while the salvage

costs of each of the items at the end of their period of use, has to be subtracted:

$$\begin{aligned}
TAV &= \sum (AC - SV) \\
&= (25200 - 5000) + (2880 - 1000) \\
&= 22080,
\end{aligned} \tag{4.11}$$

where TAV is the total assets value, AC – assets cost, SV – salvage value. In order to get the yearly wear, divide total asset value by the period of use of assets, being 3 years.

$$\begin{aligned}
W_y &= TAV/T_{use} \\
&= 22080/3 \\
&= 7360,
\end{aligned} \tag{4.12}$$

where W_y is the wear per year, TAV – total assets value, T_{use} – period of use. Relation (4.12) included tangible assets which will last for 5 years and intangible assets which last only one year. The initial value of assets in MDL was

$$\begin{aligned}
W &= W_y/D_y \cdot T_p \\
&= 7360/365 \cdot 120 \\
&= 2420,
\end{aligned} \tag{4.13}$$

4.4.3 Product cost

At this point, it is time to compute the product cost which includes direct and indirect expenses, salary expenses and wear expenses as shown in Table 4.8.

Table 4.8– Total Product Cost

Expense type	Sum (MDL)	Percentage (%)
Direct expenses	830	0.57
Intangible expenses	9400	6.45
Salary expenses	133100	91.3
Asset wear expenses	2420	1.66
Total product cost	145750	100

4.4.4 Economic indicators and results

At the moment, all the expenses for the development of the project were computed. Now is time to consider how the application should be included on the market. As mentioned above, the target group which is meaningful for the given project represent the academic infrastructures, including teachers and students. So, as starting investors, should be considered universities' administrations. There is no optimal price established for the initial project, so a indicator of 25% on top of the production cost will be used, in order to determine what the real cost the application should have.

$$\begin{aligned} GP &= C_{total}/N_{cs} + P_p \\ &= 145750/100 + 0.25 \cdot 1457.5 \\ &= 1822, \end{aligned} \tag{4.14}$$

where GP is the gross price, C_{total} – total product cost, N_{cs} – number of copies sold, P_p – chosen profit percentage. This is not the price of the end product, since it is necessary to add sales tax (VAT), which represents 20% and is added to the gross price.

$$\begin{aligned} P_{sale} &= GP + TX_{sales} \\ &= 1822 + 0.2 \cdot 1822 \\ &= 2186.4, \end{aligned} \tag{4.15}$$

where P_{sale} is the sale prices including VAT, GP – gross price, TX_{sales} – sales tax. The net income is computed by multiplying gross price and the number of expected copies to be sold, which will be

$$\begin{aligned} I_{net} &= GP \cdot N_{cs} \\ &= 1822 \cdot 100 \\ &= 182200, \end{aligned} \tag{4.16}$$

where I_{net} is the net income, GP – gross price, N_{cs} – number of copies sold. Moreover it is necessary to compute the gross and net profit. The indicators are GPr – gross profit and NPr – net profit.

$$\begin{aligned} GPr &= I_{net} - C_{production} \\ &= 182200 - 145750 \\ &= 36450 \\ NPr &= GPr - 15\% \\ &= 36450 - 15\% \\ &= 30983, \end{aligned} \tag{4.17}$$

where I_{net} is the net income, $C_{production}$ – cost of production. The profitability indicators are C_{profit}

– cost profitability, S_{profit} – sales profitability computed in MDL.

$$\begin{aligned}C_{profit} &= GPr/C_{production} \cdot 100\% \\&= 36450/145750 \cdot 100\% \\&= 25\% \\S_{profit} &= GPr/I_{net} \cdot 100\% \\&= 30983/182200 \cdot 100\% \\&= 17\%.\end{aligned}\tag{4.18}$$

4.5 Economic conclusions

After analyzing the given project from economical point of view, several conclusions were made. First of all, making such a research gave the possibility to understand better which are the strengths and the weaknesses of the system and where additional attention should be paid in order to avoid unforeseeable external factors that could create unexpected problems. Several other indexes were computed such as direct and indirect expenses, tangible and intangible resources, worker remuneration, etc. As an observation, the biggest expense would be the salaries paid to employees and future improvements for maintaining the application up to date. Now, the next step is to find suitable clients, that would use the application and, as a result, would suggest different improvements for the good functioning of the system. Finding a big constant auditory that will actively use the application would represent a big step into the future of the application, guaranteeing an economic profit and great perspectives for upcoming versions.

5 Platform usage

After finishing of first version of application after some months was observed usage of the platform by some news websites: anticoruptie.md, agora.md, deschide.md and others.

For example some articles by the journalistic investigation center anticoruptie.md:

- "Achizițiile CNA // Șase mașini Skoda Rapid, cumpărate cu 1,5 milioane de lei".

In this article platform is used to show total sum of acquisitions closed by "Daac Auto SRL" company in period of 2015-2016, also to show all institutions that closed those tenders with that company and the amount of money spent. Under that information is shown an image that describes hierarchy of founding and ownership of that company. Check Figure 5.1

Daac Auto SRL este fondată de alte două companii, Daac Hermes SA și Daac-Autotest SRL. **Daac Hermes SA** aparține omului de afaceri Vasile Chirtoca, ales local în Consiliul Municipal Chișinău din partea PCRM. La rândul ei, **Daac-Autotest SRL** a fost fondată de Daac Hermes SA. **Platforma Open Money arată că** pe parcursul anilor 2015 și 2016, Daac Auto SRL a încheiat contracte în valoare de zeci de milioane de lei cu Inspectoratul General al Poliției, Procuratura Generală, Cancelaria de Stat, Ministerul Afacerilor Interne, Agentia Medicamentului și Dispozitivelor Medicale etc. De exemplu, valoarea contractului cu Inspectoratul General al Poliției a fost de aproape cinci milioane de lei. Baza de date Open Money mai arată că CNA a încheiat contracte cu Daac Auto SRL și la sfârșitul anului 2015, în valoare de 2,1 milioane de lei.

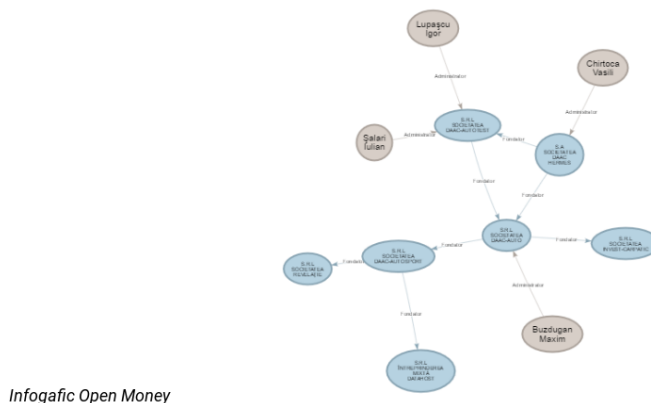


Figure 5.1 – Part of anticoruptie.md's article - "Achizițiile CNA // Șase mașini Skoda Rapid, cumpărate cu 1,5 milioane de lei"

- "Proprietarii site-urilor de știri și interesele pe care le promovează".

In this article platform is used to show real founders of a website of news "Noi.md". This website was founded by the "MLD Media SRL" company, but the company is founded directly by some persons and companies. In this case platform is used to identify indirect founders that stay from behind of direct founding companies, using foundation hierarchy of company feature in Figure 5.2 is very simple to find them all. One of these persons is "Vasile Chirtoca" - politician from PCRM. In result was found a indirect relation between a politician and website of news.

[illegible]

Deși a mediatizat detaliat campania electorală pentru prezidențiale, abordând declarațiile tuturor candidaților, în turul doi de scrutin s-a remarcat favorizarea evidentă a lui Igor Dodon și defavorizarea Maiei Sandu, remarcă experții în raportul de monitorizare al API și CJJ: „Astfel, în comentariile și alte articole de opinie, Maia Sandu a fost

[illegible]

65

Conclusions

After release of project platform was already used by some news websites like anticoruptie.md, agora.md and others in about 5-10 investigation articles mentioned in section ahead.

Main problem in implementing of project was quality of data exposed by government website, every type of data has lack of ID, so linking between entities occurred by name, there comes another problem, the names from different XLS files may differ, for example in one file "Universitatea tehnica din Moldova" but in another "Universitatea tehnica a Moldovei", so was applied some additional techniques for similarity matching like "levenshtein distance" and others. Also another problem in those data is unstable schema, older and new data differ by structure in more versions that creates.

Next major feature in application is to add versioning of graph states in time, for that moment is only the state for last quarter data.

From technological point of view in this project were many challenges. First problem was decision of general architecture for system. This platform is form-intensive, so it is ideally suited for being built as single page web application. The main idea compared to other more traditional server-side architectures is to build the server as a stateless reusable REST service. In such a way project take advantage of decoupling client part from server, which means a separation between development teams for each part and a faster implementation because of focusing on separate things with intersection just in synchronization of interface of communication (REST API). For client part we had to choose between two frameworks: AngularJS and Aurelia, both are top used frameworks, but client is part with low priority attention, so decision was based on the fact that AngularJS is more generalist with overhead of configuration stuff, but with Aurelia development occurred much faster. Part with a higher priority was server. From this part decision was first of all around which database management system to use for project's purposes. As earlier was discussed that the key point of the project is representation of all entities (persons, companies institutions) and relations (administration, founding, acquisitions between company and institution) in form of graph model, so i decided to choose a NoSQL DBMS of graph type that supports that graph model in persistent context. Such a technology has well defined things like storage management, performance, specific algorithms in its own engine for graph model, so it make much easier to implement business logic for features related to data processing in graph context. Decision was between two implementations of such DBMS type: OrientDB and Neo4j. In result the obvious winner for my purposes was OrientDB for next reasons:

- OrientDB supports Multi-Model concept (graph, documents, objects, key-values), Neo4j supports only graph model
- Performance - According to an independent benchmark by Tokyo Institute of Technology* and IBM Research*, OrientDB is 10X faster than Neo4j* on Graph operations among all the workloads
- Available features in Community Edition are more limited in Neo4j
- Declarative way of interaction in OrientDB is SQL like with graph related extension, which is more familiar than Neo4j's "Cypher" query language which is totally a new kind of language

for me, so need more time to learn it.

As server-side application language does not matter so much, for much of them exists driver from OrientDB vendor. As i am familiar with Java i have choosen it. From frameworks i have choosen ones from most popular ecosystem for enterprise solutions - "Spring" stack, Spring Core as dependency injection library and Spring MVC as web layer for request intercepting. Reason of course is large community support.

References

- 1 Craig Walls, *Spring in action*, <https://www.manning.com/books/spring-in-action-fourth-edition>, (April 4 2017)
- 2 Data & Object Factory, LLC., *.NET Design patterns*, <http://www.dofactory.com/net/design-patterns>, (April 6 2017)
- 3 Shannon Griswold, *Dependency Injection Options for Java*, <https://keyholesoftware.com/2014/02/17/dependency-injection-options-for-java/>, (April 6 2017)
- 4 Alef Arendsen, *Setter injection versus constructor injection and the use of @Required*, <https://spring.io/blog/2007/07/11/setter-injection-versus-constructor-injection-and-the-use-of-required/>, (April 8 2017)
- 5 oodesign.com, *Factory Pattern*, <http://www.oodesign.com/factory-pattern.html>, (April 10 2017)
- 6 Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, USA, Addison-Wesley, 1994, chapters 6, pages 416
- 7 tutorialspoint.com, *Design Patterns - Prototype Pattern*, https://www.tutorialspoint.com/design_pattern/prototype_pattern.htm, (April 11 2017)
- 8 Pivotal Software, *Request, session, global session, application, and WebSocket scopes*, <https://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#beans-factory-scopes-other>, (April 20 2017)
- 9 Pivotal Software, *Spring framework*, <https://projects.spring.io/spring-framework/>, (April 22 2017)