**Universitatea Tehnică a Moldovei**

**Application for finding domains of rounding the clients to the service points**

**Aplicație pentru determinarea domeniilor de arondare a clienților la punctele de deservire**

**Student: Cretu Dumitru**

**Coordonator: Corlat Sergiu,**

**Lector universitar**

**Chişinău 2021**

**MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII**

**MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Admis la susținere**

**Șef de departament:**

**Fiodorov I. dr., conf.univ.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**„\_\_\_”**  *mai* **2021**

**Aplicație pentru determinarea domeniilor de arondare a clienților la punctele de deservire**

**Proiect de licență**

**Student:\_\_\_\_\_\_\_\_\_\_\_\_\_ Cretu Dumitru, FAF-172**

**Coordonator: \_\_\_\_\_\_\_\_\_\_\_\_\_ Corlat Sergiu, Lector universitar**

**Consultant: \_\_\_\_\_\_\_\_\_\_\_\_\_ Zaharia Gabriel, asist. univ.**

**Consultant: \_\_\_\_\_\_\_\_\_\_\_\_\_ Catruc Mariana, lect. univ.**

**Chişinău 2021**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Programul de studii Ingineria Software**

**Aprob**

**şef de departament**

**Fiodorov Ion, dr.conf.univ.**

**„06” octombrie 2020**

**CAIET DE SARCINI**

**pentru proiectul de licenţă al studentului**

**Cretu Dumitru**

*(numele şi prenumele studentului)*

**1. Tema proiectului de licenţă Aplicație pentru determinarea poziționării optimale a locației(ilor) de deservire confirmată prin hotărârea Consiliului facultăţii nr. 2 din „06” octombrie 2020**

**2. Termenul limită de prezentare a proiectului de licență „21” mai 2021**

**3. Date iniţiale pentru elaborarea proiectului de licență** *Sarcina primită de la coordinator*

**4. Conţinutul memoriului explicativ**

*1 Domain of Study Analysis*

*2 Modeling*

*3 Development*

*4 Economical research*

*5 Conclusion*

**5. Conţinutul părţii grafice a proiectului de licență**

*UML diagrams, fragments of maps from the Yandex Maps service, Gantt chart, histograms, Voronoi diagrams, function graphs, screenshots of programs*

**6. Lista consultanţilor:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Consultant** | **Capitol** | **Confirmarea realizării activităţii** | |
| **Semnătura consultantului (data)** | **Semnătura studentului (data)** |
| Zaharia Gabriel | Standarde tehnologice |  |  |
| Catruc Mariana | Controlul calității |  |  |

**7. Data înmânării caietului de sarcini 06.10.2020**

**Coordonator** *Corlat Sergiu*  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*semnătura*

**Sarcina a fost luată pentru a fi executată de către studentul**

*Cretu Dumitru*  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

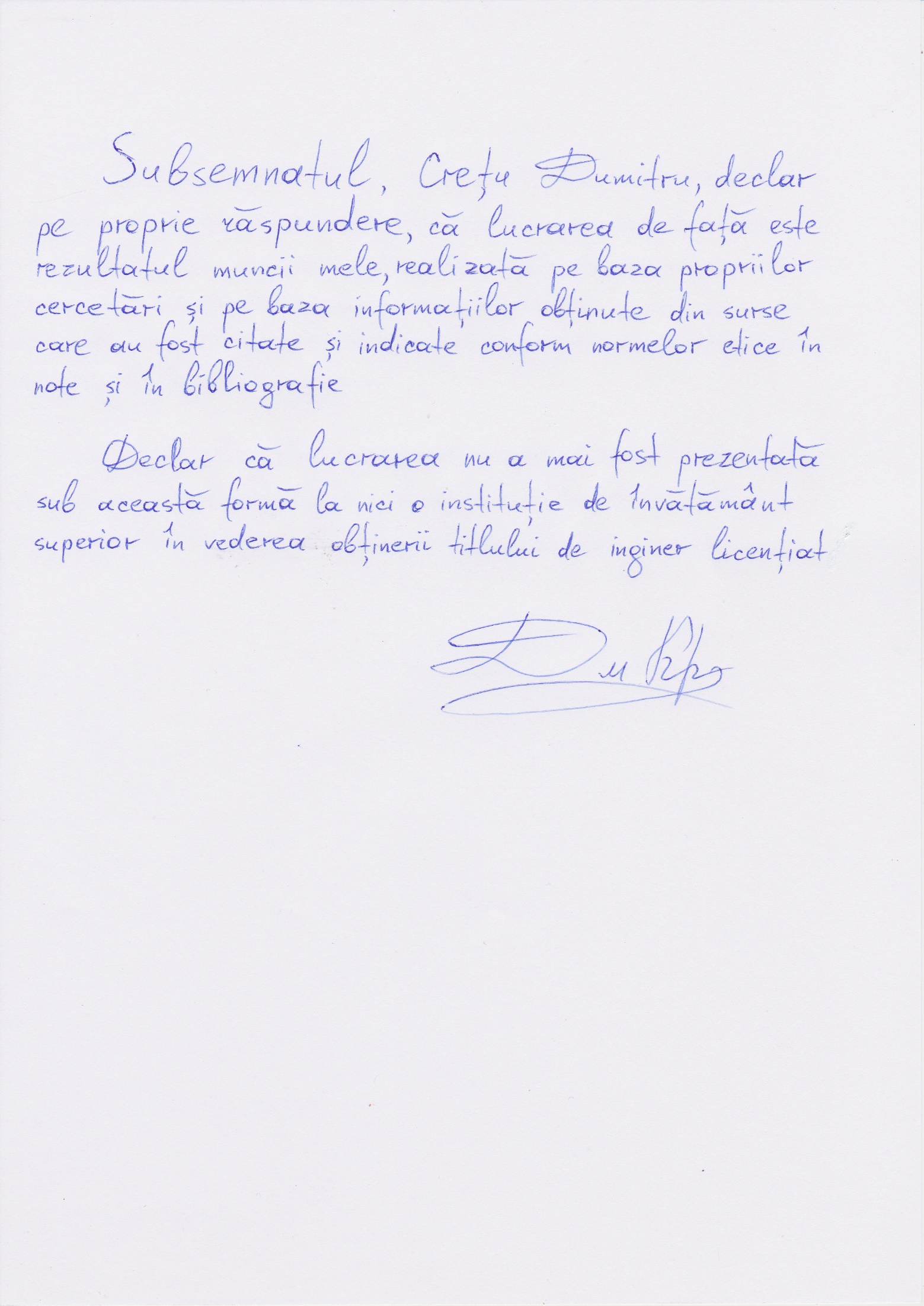
*Semnătura, data*

**PLAN CALENDARISTIC**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr. crt.** | **Denumirea etapelor de elaborare/proiectare** | **Termenul de**  **realizare a etapelor** | **Nota** |
| *1* | Elaborarea sarcinii, primirea datelor pentru sarcină | *01.09.2020 – 31.10.2020* | *5%* |
| *2* | Analiza domeniului de studiu | *01.11.2020 – 14.01.2021* | *20%* |
| *3* | Proiectarea sistemului | *15.01.2021 – 09.03.2021* | *15%* |
| *4* | Realizarea sistemului | *10.03.2021 – 15.04.2021* | *35%* |
| *5* | Descrierea sistemului | *16.04.2021 – 27.04.2021* | *10%* |
| *6* | Testarea sistemului | *28.04.2021 – 10.05.2021* | *10%* |
| *7* | Finalizarea proiectului | *11.05.2021 – 21.05.2021* | *5%* |

**Student**  *Cretu Dumitru ( )*

**Coordonator de proiect de licență**  *Corlat Sergiu ( )*

**

**ADNOTARE**

*Numele și prenumele autorului: Crețu Dumitru.*

*Titlul tezei: Aplicație pentru găsirea domeniilor de rotunjire a clienților la punctele de servicii.*

*Cuvinte cheie: punct de serviciu, diagramă Voronoi, locus, site, zonă de acoperire.*

*Baza teoretică utilizată în teză: diagrama Voronoi, distanța euclidiană, distanța Manhattan, metoda perpendiculară, metoda Fortune, algoritmul recursiv, teoria mulțimilor, polinoamele de gradul I.*

*Structura de lucru: Introducere, Analiza domeniului, Modelare, Dezvoltare, Analiza economică, Concluzie, Bibliografia*

*Scopul tezei: dezvoltarea unui program de distribuție a clienților pe puncte de servicii.*

*Materiale utilizate: limbaj de programare Java, bibliotecă grafică Swing, Ediție comunitară Intellij IDEA.*

*Semnificație practică și teoretică: dezvoltarea de software pentru analiza cartografică a zonelor de acoperire a punctelor de servicii și a distribuției clienților.*

*Rezultatul tezei: un prototip al unei aplicații cartografice pentru distribuția optimă a zonelor de acoperire de către punctele de serviciu pe zona indicată a hărții cu transformarea coordonatelor între coordonatele euclidiene și coordonatele geografice în versiunile controlate prin linia de comandă și printr-o interfață grafică cu utilizatorul.*

**ANNOTATION**

*Last name and first name of the author: Cretu Dumitru.*

*Title of thesis: Application for finding domains of rounding the clients to the service points.*

*Keywords: service point, Voronoi diagram, locus, site, coverage area.*

*The theoretical basis used in the thesis: Voronoi diagram, Euclidean distance, Manhattan distance, perpendicular method, Fortune's method, recursive algorithm, set theory, polynomials of the first degree.*

*Work structure: Introduction, Domain analysis, Modeling, Development, Economic analysis, Conclusion, Bibliography*

*The purpose of the thesis: development of a program for the distribution of customers by service points.*

*Materials used: Java programming language, Swing graphic library, Intellij IDEA Community Edition.*

*Practical and theoretical significance: development of software for cartographic analysis of coverage areas of service points and customer distribution.*

*The result of thesis: a prototype of a cartographic application for the optimal distribution of coverage areas by service points on the indicated area of the map with coordinate transformation between Euclidean coordinates and geographic coordinates in versions controlled through the command line and through a graphical user interface.*

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**FACULTATEA CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ**

**DEPARTAMENTUL INGINERIA SOFTWARE ȘI AUTOMATICĂ**

**PROGRAMUL DE STUDII INGINERIA SOFTWARE**

**AVIZ**

la proiectul de licenţă

**Titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Studentul(a)  *Cretu Dumitru*  grupa\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Actualitatea temei: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2. Caracteristica proioectului/tezei de licenţă:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**3. Analiza prototipului: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**4. Estimarea rezultatelor obţinute: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**5. Corectitudinea materialului expus: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**6. Calitatea materialului grafic: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**7. Valoarea practică a proiectului/tezei: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**8. Observaţii şi recomandări: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**9. Caracteristica studentului şi titlul conferit : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Lucrarea în forma electronică corespunde originalului prezentat către susținere publică.***

**Coordonatorul proiectului de licenţă** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TABLE OF CONTENTS**

INTRODUCTION 14

1.1. Business problem 16

1.2. Description of the domain 17

1.2.1. Options for solving the problem 19

1.2.2. The goals/objectives, mission of the system 23

1.2.3. Comparison of the developed solution with the existing ones 24

1.2.3. The choice of technologies for solving the problem 25

1.3. Technical specifications of the program 25

1.3.1. System resource requirements 25

1.3.2. Functional and non-functional requirements 26

Conclusion 28

2. MODELLING 29

2.1. User interaction models and diagrams 29

2.1.1. Use-case diagram 29

2.1.2. Sequence diagram 30

2.2. Inner structure models and diagrams 32

2.2.1. Component diagram 32

2.2.2. Deployment diagram 33

2.2.2. Class diagram 35

Conclusion 37

3. DEVELOPMENT 38

3.1. Algorithms for constructing a Voronoi diagram 38

3.1.1. Pixel-by-Pixel method 38

3.1.2. Perpendicular method with half-planes intersection 39

3.1.3. Perpendicular method with polygon clipping 42

3.2. Additional features 43

3.2.1. Yandex Maps Static API 43

3.2.2. JSON sites storage 44

3.2.3. Graphic User Interface 45

3.3. Performance test 47

3.3.1. Opportunities to improve performance 48

Conclusion 49

4. ECONOMIC RESEARCH 51

4.1. Business case 51

4.2. Target audience/market segment 52

4.2.1. Customer avatar 53

4.3. SWOT analysis 54

4.4. Time management and estimations 56

4.4.1. Gantt chart 56

4.4.2. Scrum project management 61

4.5. Costs and profit estimation 61

4.5.1. Labor costs 61

4.5.2. The costs of opening and monthly maintenance of the company 62

4.5.3. Marketing costs 63

4.5.4. Profit estimations 64

4.5.5. Further development with financial success and investment 64

Conclusion 65

CONCLUSIONS 67

BIBLIOGRAPHY 68

ANNEXES 73

**TABLES LIST**

Table 1.1………………………………………………………………………………………………... 24

Table 3.1………………………………………………………………………………………………... 47

Table 3.2………………………………………………………………………………………………... 47

Table 3.3………………………………………………………………………………………………... 47

Table 3.4………………………………………………………………………………………………... 48

Table 4.1………………………………………………………………………………………………... 51

Table 4.2………………………………………………………………………………………………... 55

Table 4.3………………………………………………………………………………………………... 60

Table 4.4………………………………………………………………………………………………... 62

Table 4.5………………………………………………………………………………………………... 63

Table 4.6………………………………………………………………………………………………... 63

**FIGURES LIST**

Figure 1.1………………………………………………………………………………………………… 16

Figure 1.2………………………………………………………………………………………………… 19

Figure 1.3………………………………………………………………………………………………… 20

Figure 1.4………………………………………………………………………………………………… 20

Figure 1.5………………………………………………………………………………………………… 21

Figure 1.6………………………………………………………………………………………………… 22

Figure 1.7………………………………………………………………………………………………… 23

Figure 2.1………………………………………………………………………………………………… 30

Figure 2.2………………………………………………………………………………………………… 32

Figure 2.3………………………………………………………………………………………………… 33

Figure 2.4………………………………………………………………………………………………… 34

Figure 2.5………………………………………………………………………………………………… 34

Figure 2.6………………………………………………………………………………………………… 35

Figure 2.7………………………………………………………………………………………………… 36

Figure 3.1………………………………………………………………………………………………… 39

Figure 3.2………………………………………………………………………………………………… 39

Figure 3.3………………………………………………………………………………………………… 41

Figure 3.4………………………………………………………………………………………………… 41

Figure 3.5………………………………………………………………………………………………… 42

Figure 3.6………………………………………………………………………………………………… 42

Figure 3.7………………………………………………………………………………………………… 44

Figure 3.8………………………………………………………………………………………………… 45

Figure 3.9………………………………………………………………………………………………… 45

Figure 3.10……………………………………………………………………………………………….. 46

Figure 4.1………………………………………………………………………………………………… 56

Figure 4.2………………………………………………………………………………………………… 57

Figure 4.3………………………………………………………………………………………………… 57

Figure 4.4………………………………………………………………………………………………… 58

Figure 4.5………………………………………………………………………………………………… 58

Figure 4.6………………………………………………………………………………………………… 59

Figure 4.7………………………………………………………………………………………………… 59

Figure 4.8………………………………………………………………………………………………… 60

Figure 4.9………………………………………………………………………………………………… 61

# **INTRODUCTION**

At the time of this writing, the coronavirus pandemic is raging around the world, prompting many companies to switch to remote work with remote customer service. Due to these changes, companies are forced to deliver their goods and services to customers, which is a costly, complex, and difficult to organize process.

Despite the obstacles, any company strives to get as much income as possible, serving as many customers as possible. Considering the need to deliver goods or services, the distance of clients from the point of service becomes a determining factor for the speed of service for each client.

This problem is significant not only in business. The speed at which government agencies such as the police and ambulance services can help often depends on the distance between service representatives and people in need. Minimizing the time of arrival in such a case affects the provision of assistance or the rescue of more people.

The main principle of determining the optimal point of service in most map services is to analyze their distance from the specified location by distance or time of arrival, which leads to the need for constant recalculation of results when specifying a new location or changing existing services. Given the need for frequent recalculation, point management, and the ability to visualize optimal service areas, there is no effective solution to this problem.

To solve these issues, an attempt was made to implement an algorithm for optimizing the coverage area of ​​service points as a solution developed in the work over the current thesis. This algorithm will allow any user to determine the optimal (best) point of service for the clients depending on their location, find optimal coverage areas, which will lead to a reduction in the delivery time of goods / services, and minimize the response time to a request. In government services, this system will allow to quickly respond to incoming requests and resolve them faster.

An effective algorithm for calculating the optimal service areas can become a significant solution, interesting both from the point of view of its mathematical implementation, considering need to present this solution as an algorithm and to implement it as a program, and from the economic point of view, by considering it as a possible sold product.

The problem involves the analysis of models for calculating the optimal areas of service points, determining their computational complexity, the implementation of additional capabilities to consider the possible state of the point of interest, consideration of third-party solutions on the market, which boils down to the following objectives:

* identification of the problem, modeling the problem solution;
* selecting methods and mathematical algorithms for solving the problem;
* selecting tools for the development of the IT-realization of the problem solution;
* description of processes, situations, models associated with the problem;
* research economic situation related to this problem;
* development of the application, setting up a working prototype;
* collection of data required for developing the application
* perform prototype tests;
* application launch.

The sequence used in elaboration of this project is problem analysis, mathematical modeling and mathematical approach of solving the problem, transformation of estimated theoretical basis into algorithms, computer modelling (program implementation), application testing.

This thesis consists of several chapters, each of which has its own role:

* Introduction - a description of the topic of this thesis, an indication of the structure of the document, the setting of tasks within the limits of this work;
* The first chapter - theoretical aspects of solving the problem, mathematical aspects of solving the problem, studying the issue and its decomposition;
* The second chapter is the modeling of the program, its structuring, decomposition of the program into components and their analysis;
* The third chapter is the development of the program, the description of the implemented algorithms, the demonstration of the work of the program, the creation of a working prototype;
* The fourth chapter is an analysis of the project from an economic point of view, considering the project as a startup;
* Bibliography - Sources used to write the thesis (books and web pages).

**1. DOMAIN OF STUDY ANALYSIS**

The current chapter describes in detail the issues and problems solved during the work over the current thesis, determining the significance of the project for the market and for people in general, the alternatives present on the market, their impact on the problems raised and issues with their direct comparison with the software developed to solve these problems. This chapter will also demonstrate methodologies for solving the described problems, setting goals and objectives for the developed program, identifying development tools, setting specifications and requirements for the developed program.

## **1.1. Business problem**

The pace of modern life requires quick customer service regardless of the goods / services provided. In business, less time spent on customer service leads to the ability to serve more customers in a given period of time, increases customer satisfaction and increases trust in the company [1]. Less time to help emergency services will save more lives or provide more needed assistance.

Private companies are actively expanding the sphere of delivery of goods / services [2], switching to systems of online interaction with customers [3] [4]. With the transition to remote service with the subsequent delivery of goods / services, optimization of routes, delivery methods and regions served is becoming an important issue to meet the time requirements of customers.

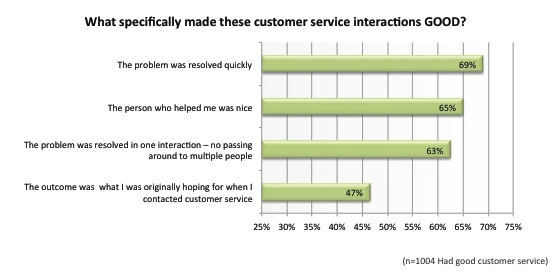


Figure 1.1 - Zendesk’s research survey results (taken from source [5])

According to statistics and figure 1.1, the most important customer service requirement is speed [5] [6]. The remoteness of the service point most of all reduces the speed of service due to the time spent on the delivery of the request / product / service or the arrival of the client. The issue of the remoteness of service points is relevant for both traditional forms of business (gas stations, restaurants, shops) and modern ones (servers, data centers) [7].

Mapping services like Yandex Maps or Google Maps allow determination of the shortest path between two points and allow the average user to determine the closest service point of interest, depending on the category. Despite the applicability of such a solution for individuals, legal entities suffer from many shortcomings of these services, such as the impossibility of operational management of service points, due to time delays on the part of map services to verify requested changes, lack of visualization of coverage areas of service points, delays in determining the optimal paths with distance between points. These factors negatively affect the efficiency of traditional forms of business.

IT companies working with web applications or web services in many cases provide their services in "location transparency" mode, when the user can be redirected to the nearest server accessible to the user with the requested capabilities to minimize delays in working with the application or service. To determine the nearest server, a company can use an algorithm for checking the distance from all available servers to a given client, but such an algorithm will have delays and the need to recalculate each time a new client connects. If the central server that distributes connections across servers has diagrams describing the coverage areas of each server, this process can be automated and reduced to minimum delays, increasing the speed of operation and response of these services.

To improve the efficiency of delivery and resource allocation processes, companies need logistics planning. A program for determining optimal coverage areas could be an effective tool for logistic analysis, as it will provide an effective means for identifying the most profitable suppliers of resources, supply chains, and so on.

This program may be of interest to urban planning enterprises for planning the distribution of buildings of a certain type in the city, defense institutions in view of the possibility of using it to determine the covered area of ​​protection or defense by military equipment, government agencies such as the Ministry of Health for the allocation of resources depending on the area covered by some medical institution, and so on. The areas of application of this program are wide and can have a positive impact on the work of the mentioned institutions.

Due to the market demand for the implementation of the point of service coverage area identification tool, it was decided to develop a corresponding program. The program will be based on determining the coverage area of ​​the service point, depending on the location of this point and all others on the requested area of ​​the map, demonstrate all changes in real time for ease of point management process, visualize the changes being made as a diagram convenient for visual analysis and demonstration of the results of the distribution of service points, plot the diagram over the specific area on the map. Based on the previously considered factors, it can be assumed that legal entities and corporate clients are interested in purchasing the developed software and the impact of this development on the market will be favorable.

## **1.2. Description of the domain**

The sequence of delivery of products or services most often follows a simple scheme: the company receives the order, determines the location of the client, executes the delivered order, and sends this order to the client. If the company has several service points, the definition of the nearest service point is added to this chain of actions, the execution and delivery of the order is assigned to the found point. To determine the optimal (closest or best) point of service, an enterprise can use the services of a logistics operator or mapping services.

One of the popular mapping services is Google Maps. It works according to the GIS framework, which is based on considering a route from one point to another as a set of points that are connected to each other using "paths" (the principle of operation is similar to directed graphs). If the route is successfully laid from the starting point to the destination point, the system will check if the route contains nested points. If there are any, the system will sum up the distances between the points, resulting in the total distance of the route [8] [9] [10]. Google Maps provides the ability to determine the route and distance between two points in one request, as a result of which to check the remoteness of several service points several requests will be required. The company, if it wants to save money, can go for a trick - write a script that allows based on the Google Maps query system to determine the distances to service points and, based on the information received, find the nearest service point (it is also possible to save the route). Such a solution will suffer from limits on the Internet connection, delays in the delivery of requests over the Internet, the constant need to recalculate the distance for each new location of the client, the inability to quickly adjust the service points on the map and from the lack of visualization of the coverage areas of the service points.

An alternative option for determining the optimal service point can be seen in the solution on the local market, the "straus.md" service (food delivery service). A careful study of the service will show the limitation of the service operation within the city of Chisinau with the distribution of service areas based on the area in which the client is located. A circular area is formed around the specified location, which marks restaurants that fall within its area as the best options [11]. The calculation allows user to find the closest set of objects, but still does not allow building an optimal coverage area for service points and requires recalculation for each new client location.

A full-fledged competitor providing services for the construction of coverage areas for chosen points can be considered the ArcGIS platform [12], which consists of a wide range of cartographic tools available on a paid basis. This platform has been developing since 1999 [13] and provides extremely wide opportunities for its users. This platform will be able to solve the tasks set for this thesis; however, it is worth noting several important aspects that do not allow it to be considered as an uncontested solution.

The platform is distributed based on the payment of an annual subscription to the services of this service [14]. For individuals or students, the cost of using this service is $ 100, the cost for legal entities is set on an individual basis based on the requested services. The platform is aimed at solving a variety of cartographic operations, due to which the user will be forced to understand the complex structure of the platform and its set of tools, which is time-consuming and resource-intensive [15]. In the figure 1.2 is presented example of graphical user interface of ArcGIS Pro application.

The ArcGIS platform is a complex tool, which will require a detailed study of the principles of work from the user. As an example, the analysis of the service area consists of many aspects and settings that the user will have to specify in order to generate the appropriate graphs and tables. The system provides a wide range of tools, but for simple analysis this platform is a cumbersome solution.

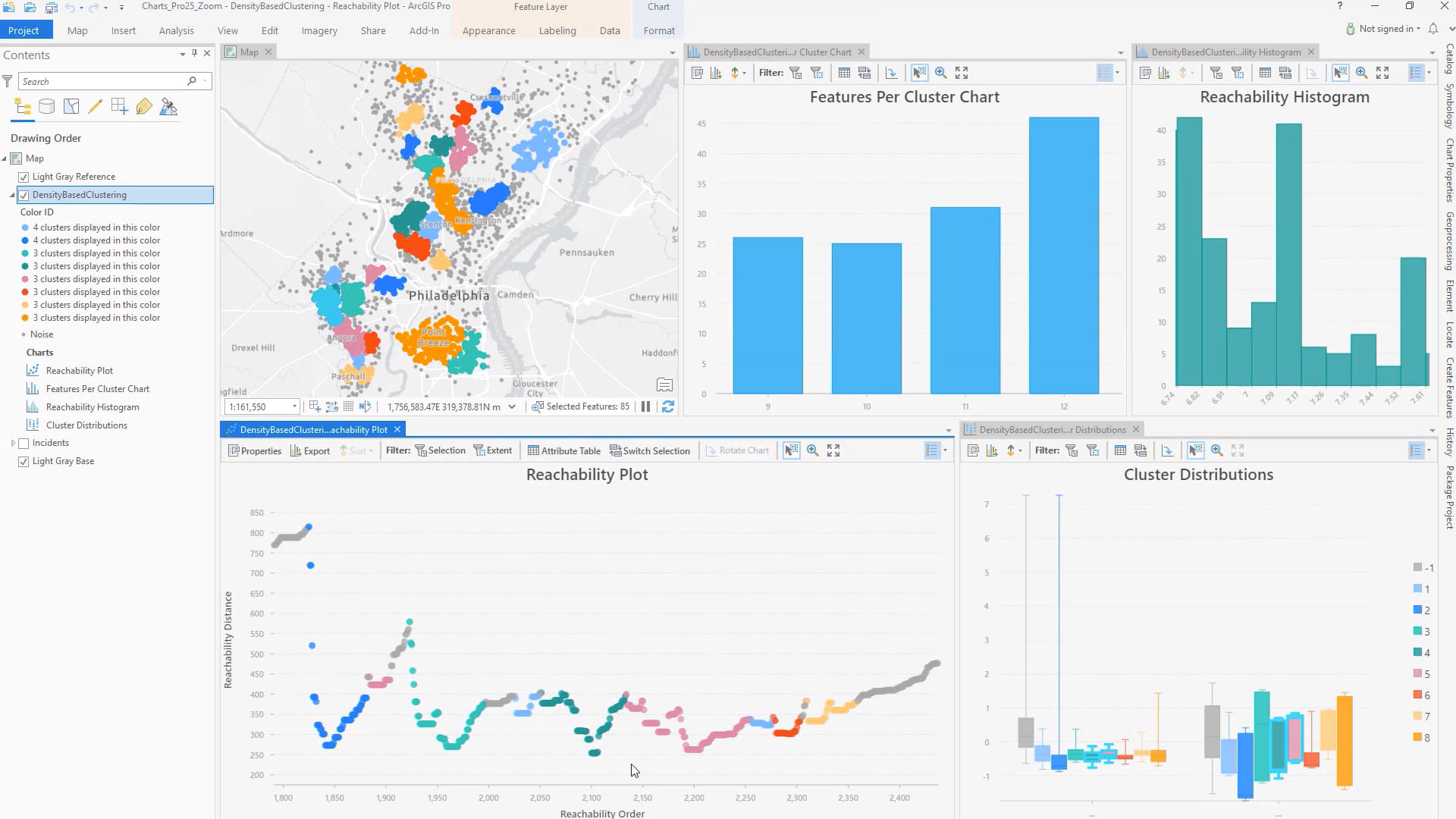


Figure 1.2 - screenshot of working ArcGIS Pro (taken from source [16])

Despite the availability of solutions on the market, it is worth noting either the lack of effective solutions, or their increased cost with the complexity of use. Considering the importance of the problem, the implementation of software without the above-mentioned disadvantages is necessary, favorably affecting the situation in the described domain. The economic side of the development process, economic analysis of the market, accurate audience definition, and other economic aspects will be demonstrated in the fourth chapter.

### **1.2.1. Options for solving the problem**

To begin with, it is important to decide what the "optimal service area" is. An optimal service area is a plane where all points within which will be closest to a given service point than to any other. This formulation is extremely close to the mathematical definition of loci - geometric areas, all points of which are closer to the main point of a given area than to any other area [17]. Main point of this area is called site, meaning that in the following, the term "service point" may be replaced by the term "site" and the term "service area" by "locus".

The construction of the locus of each site leads to the creation of a Voronoi diagram. The Voronoi diagram is a region divided into many loci [18]. The construction of a Voronoi diagram is possible using one of four algorithms. Each will be reviewed separately for complete understanding [19].

The easiest way to construct a Voronoi diagram is to check pixel-by-pixel whether a point belongs to a certain locus. This method is the simplest to implement, the simplest to understand, but gives the worst results in terms of execution time - the computer will have to go through each pixel separately and individually determine the nearest point for each, which implies a bulkhead of all points on the plane. Thus, the time complexity of the algorithm will be

(1.1)

This technology is applicable when it is necessary to quickly solve a problem, but the number of required calculations is high, reducing the applicability of this method (figure 1.3 presents principle of finding closest site conform this method). The worst results this method shows if there is a small points value.

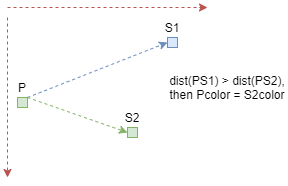


Figure 1.3 - visualization of pixel-by-pixel method, where P is representing pixel and S1 with S2 are representing sites (made by author)

How to visually verify that the point is inside the locus? Or, to rephrase, how to check if a dot is closer to one site than to another? Draw a line between the sites, draw a perpendicular in the middle of the line and check on which side relative to the perpendicular this point will be. This is how the principle of calculating the locus by the perpendicular method works. Figure 1.4 visualizes this principle.

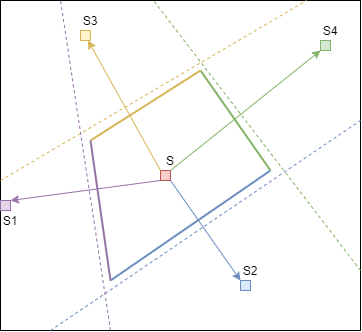


Figure 1.4 - visualization of perpendicular method, where arrows represent lines to sites, dotted lines are perpendiculars, bold lines are perpendicular parts that form current site locus (made by author)

When constructing each perpendicular, the plane is divided into two parts, into two half-planes. Therefore, drawing perpendiculars for each connector to a different site will create a set of half-planes for the current site. The intersection of all the resulting half-planes will result in the locus of this site [20].

The time complexity of the algorithm is much better than the previous one. Calculating the intersection of all half-planes for the current site will require *n log n* operations. Therefore, the calculation for all sites on the plane will be .

This method is distinguished by the ability to change and edit it in such a way that it considers some of the features of the service points. Let's say that the company has two points, one of which is able to serve customers faster and it can cover a large area. This algorithm can be modified for the system of "weights", which will determine where the perpendicular will be drawn on the connected line (in favor of a point with a large weight).

The complexity indicators of this method already allow using it to build a working solution, however, alternative algorithms need to be considered [19].

Consider Fortune's algorithm. Let's imagine that there are *n* sites on the plane. The sweeping line will move along this plane from the larger ordinate to the smaller one. Each time sweeping line encounters any site, a parabola is created, the focus of which becomes the encountered site, identifying the sweeping line as the directrix. The parabola line will consist of points equidistant from the focus and directrix. This is how the locus of this point begins to form.

When a parabola crosses another parabola, a "control point" is formed. As the sweeping straight-line shifts, the parabola becomes larger and when two control points meet (in other words, when the intersection of three parabolas appears), the vertex of the locus is formed. When a parabola begins to overlap with other parabolas, it is removed and the locus is considered complete [21]. The general principle of work is shown in the figure 1.5.

In the case when the program knows which edge belongs to which site, it is possible to find the edge of the neighboring site without additional calculations. The time complexity of such an algorithm will be *n log n*, provided that there are *n* sites and the complexity of constructing a locus is *log n* [22].

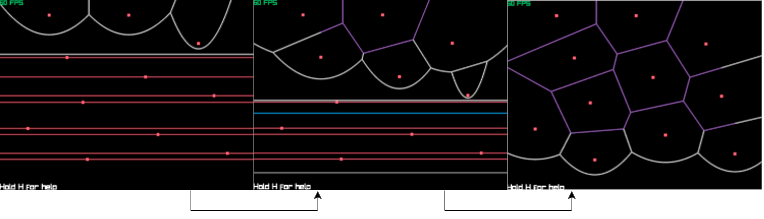


Figure 1.5 - Fortune’s algorithm visualization (taken from source [23])

The problem of Fortune's method is the need to determine the function describing the parabola to determine the intersection points of the parabolas. A parabola is given by a second-order polynomial, the calculation of which is necessary for each shift of the sweeping line, and the definition of the locus vertex will be the intersection point of three parabolas and, therefore, the intersection point of three second-order polynomials. This will require writing a system for solving second-order equations, the implementation of which can be difficult, and an efficient implementation of complex mathematical calculations.

An additional limitation comes down to the need for permanent storage in memory of all incomplete (unclosed) site loci. Any uncomplete site locus is stored inside a "coastline". This coastline should have an algorithm to record the locus of the newly discovered site and remove the completed loci. To determine the "completeness" of a drawable locus, an algorithm is required that limits the arc within either a given area, or when it collides with another arc or polygon face, which can also be a complex stage in the implementation and significantly complicate the development. The algorithm, due to its complexity, is also not subject to modifiability in order to consider the factors affecting the size of loci. Due to these limitations, it was decided to abandon the use of this method.

The last solution for constructing a Voronoi diagram is a recursive method for constructing a diagram. This technique is based on splitting the original set of sites into two parts. The program calculates the Voronoi diagram for each part separately, but here the main limitation of this diagram building algorithm appears.

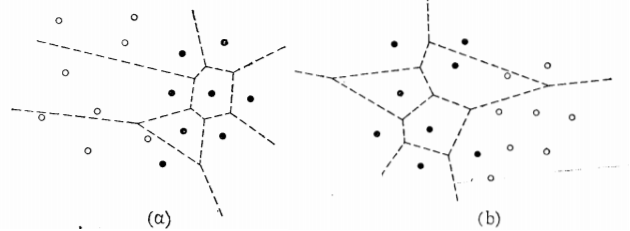


Figure 1.6 - recursive Voronoi Diagram construction (taken from source [19])

In the figure 1.6, can be seen the division of the points into regions into two parts. First, the Voronoi diagram is determined for points from the area on the right, and then it is determined for the area on the left. Due to the peculiarities of plotting, when trying to directly connect diagrams, diagrams will overlap, forming an erroneous display of the general diagram. Between the points located at the separation boundary, can be drawn a polyline consisting of pieces of dividing perpendiculars, which will overlap the overlapping diagrams and allow building the correct overall diagram.

The whole complexity of the method lies in the correct construction of this broken line. Building a straight line based on sites located near the border is impossible due to the possibility of missing a locus that is distant from the border, but affects both diagrams. An alternative algorithm assumes the definition of external sites of the subdivided areas, their connection into convex polygons, after which the upper line is constructed for the upper vertices of these polygons and the lower one for the lower vertices. Thus, the two outer perpendiculars are defined, and the inner perpendiculars are determined by refracting a straight line from "collision" with the face of the locus and thereby limiting the specified locus, requiring removal of lines that are coming to the other half side. The principle of finding broken line is presented in the figure 1.7.

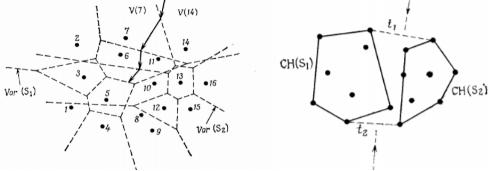


Figure 1.7 - algorithm to find broken “connection” line for separated Voronoi Diagrams (taken from source [19])

The time complexity of constructing this algorithm is *n log n*, achieved by decomposing the diagram into small components, the calculation of which is simpler, and the procedure for detecting a broken line has linear complexity. The time complexity of the algorithm is attractive, but the stage of defining the broken line excludes the possibility of effectively "parametrizing" the construction of loci (introducing new factors affecting the coverage areas of sites) to improve the algorithm in the future. For these reasons, it was also decided to abandon the current algorithm [24].

To implement the task set for this thesis, the first two methods of constructing a Voronoi diagram were used. Recursive and Fortune diagrams are complex and require more time and resources to implement. When creating a startup in which a small team of developers will work on the program, these methods are more realizable.

Additional factors that influenced the rejection of these solution options were the increased need for memory and the complexity of breaking up according to the recursive method, as well as the cumbersomeness and difficulty in modifying the Fortune method. The method of perpendiculars is not the most optimal in terms of algorithmic complexity, however, in comparison with others, its implementation will be more efficient and will meet the requirements set for the program.

### **1.2.2. The goals/objectives, mission of the system**

The main scopes of the solution provided in this thesis:

* Creation of a convenient and easy-to-use user interface that will make it easy to work with the system;
* Implementation of a system for adding, editing and deleting service points;
* Determination of the nearest (optimal) service point based on the built coverage area and the location of the requested client or destination;
* Visualization of optimal coverage areas within a given area of ​​the map
* Providing the ability to dynamically change the displayed diagram based on the changes made to visualize the movement of service points;
* The ability to indicate the location on the map for which the analysis of the optimal coverage areas will be carried out;
* Export and import of information on service points into separate files for the possibility of saving, transferring or entering into the system;
* Implementation of the program with minimal requirements for computing resources.

The mission set for the solution implemented within the solution of this thesis - providing effective tools for determining the optimal coverage areas for points in a user-friendly format with the ability to quickly determine the optimal point of service based on the requested location.

### **1.2.3. Comparison of the developed solution with the existing ones**

Despite the belonging of Google Maps to cartographic services, it is worth noting its low comparability with the software being developed and its focus on solving other problems. The situation is similar with Yandex Maps. Only ArcGIS can be considered as a full-fledged alternative, despite the inclusion of a whole system of solutions and programs for cartographic analysis by this platform.

Table 1.1 - Comparison of the developed program with alternative solutions (made by author)

|  |  |  |
| --- | --- | --- |
| **ArcGIS** | **Criteria** | **ClientOptService (Optimal Service Area Calculation)** |
| Cartographic analysis, analysis of objects located on the map, their classification, systematization, provision of tools for working with maps. | **Purpose** | Determining the optimal service point coverage area, determining the optimal service point for a specified location. |
| Reliable for cartographic analysis companies, for collecting cartographic data, for cartographic researchers. | **Reliability** | Reliable for companies engaged in optimization of delivery processes, logistics analysis, resource allocation, distribution of service points, cartographic analysis. |
| Performing mathematical, geometric, or analytical operations on map data. | **Usability** | Determining the optimal service area by a service point based on location and other service points. |
| Available for companies engaged in complex mapping research and analysis. | **Availability** | Available for companies wishing to optimize delivery processes, simplify logistics operations, and optimize resource allocation. |

The comparative table demonstrates a greater coverage, a greater number of problems covered, a greater number of opportunities provided by ArcGIS. Despite the wide applicability within the limits of the solution of the narrowly focused issue raised, the use of such a cumbersome and wide solution when it is necessary to solve only this issue is suboptimal and may turn out to be of little interest to potential customers. When setting the task to determine the coverage areas, the company will not be interested in such a wide range of tools that the ArcGIS platform provides, as a result of which the acquisition of this solution will be an overpayment for unused opportunities. Considering the high cost of this solution, a cheap and narrowly targeted analogue may turn out to be more attractive, as a result of which the developed program will be able to occupy its own niche in the market.

### **1.2.3. The choice of technologies for solving the problem**

Since the program will be aimed at legal entities using computers as the main tools for work, for the developed software, it is important to be cross-platform and be able to run regardless of the operating system used. For this reason, the Java programming language was chosen. With correct use, programs in this language can show high performance and, due to the strict class system, the readability of the code is significantly increased. Also, internal use of the garbage collector eliminates errors of memory leaks or suboptimal resource use.

To draw the graphical interface of the program and visualize the generated diagram with a map, the Swing graphical library used in Java by default will be used. This solution will provide low resource consumption due to the optimized given graphic library and will simplify the porting of the program to new versions of Java due to compatibility with the basic elements of the language. This language also supports a wide range of possibilities for working with the Internet network, which makes it the best choice for working with the Internet.

To be able to control versions of the developed software, save the source code of the program, and the possibility of further development by several people, the Git version control system is used and all the code is loaded into separate GitHub repositories (source code repositories links are presented in the end of the current thesis).

To be able to use third-party libraries in the future, the project will be developed using the Maven project management and assembly tool. The integrated development environment IntelliJ IDEA Community Edition will be used to write the code.

## **1.3. Technical specifications of the program**

The program implemented during the work over this thesis is aimed at determining the optimal coverage area of ​​the service point. The design of the coverage area should consider the factors affecting the operation of the service point and the program should interact with the user based on the data entered by the user. The coverage zones will be constructed using the Voronoi diagram.

A working prototype is being developed for use in the local economic market, which does not exclude the possibility of further development of the program for the possibility of its use abroad. The main users of the program will be legal entities and government agencies, whose activities depend on the distribution of the load between service points or for cartographic analysis of the given map area based on location and coverage areas by service points or other elements.

### **1.3.1. System resource requirements**

System resource requirements:

* Mathematical resources - calculating the optimal coverage area of ​​the service point based on the selected mathematical algorithm, algorithms for converting the coordinates of points from geographic to Cartesian for the possibility of their visual display on the resulting diagrams;
* Information resources - operating with the incoming data in the form of coordinates with a set of characteristics, getting cartographic data in image form, building the geometric areas to be operated, following architectural and design patterns;
* Linguistic resources - Java programming language, Swing graphic library;
* Design - independence of the program from the use of third-party solutions, the integrity of the program;
* Methodological - specified in the documentation of all implemented features

Administrative system requirements:

* Technical resource management;
* Management of the program and the behavior of devices with this program;
* Management of indicated service points, timeliness of making changes to the state of service points.

Requirements for technical support of the program:

* Timely correction of detected errors, inaccuracies in the work of the program;
* Support responsiveness.

### **1.3.2. Functional and non-functional requirements**

Functional requirements of the system are [25]:

1. Service point management:
   * Adding a new point;
   * Deleting a point;
   * Point editing;
   * Checking the coordinates of the service point for its presence within the specified area when creating or editing it, refusal if the check fails;
2. Controlling the graphic output of the program's results:
   * Displaying the resulting diagram (areas with distributed zones) in the form of a separate section of the window;
   * Changing the displayed diagram in real time depending on changes of the service points;
   * Changing representation mode by request of the user to be with or without map fragment background;
3. Working with a specified destination:
   * Finding the optimal service point for the given location;
   * Showing the optimal service point info;
   * Ignore location which is set outside of the given map section;
4. Working with the analyzed area:
   * Setting the analyzed area at the first start of the program;
   * Editing the location of the analyzed area;
   * Checking for the presence of all previously entered service points within the specified area, drop all points that are not entering in the new area;
5. Working with the file system:
   * Saving the set points of service and their parameters in a separate file;
   * Reading set of service points from the file;
6. User interface:
   * Dividing the window into two vertically divided parts, the left part of which displays the result of calculating the Voronoi diagram, and the right part displays the information entered by the user, forms for interacting, the outputs of the program;
   * Availability of forms for entering a new service point, editing, showing info and deleting an old one, setting new map area center;
   * Availability of a form to indicate the location for which it is necessary to determine the optimal point of service;
   * Output of the found optimal service point in the form of a log;
   * Ability to switch between modes of representation of the found Voronoi diagram;

Non-functional requirements:

1. Performance:
   * The ability to create a Voronoi diagram with its visualization for 80-120 service points in less than 1 second;
   * The ability to create a Voronoi diagram without visualization for 80-120 service points in less than 0.5 second;
2. Scalability:
   * Splitting the program into a system of modules that can be implemented in distributed systems;
   * Having both graphical interface and command-based interfaces for work with the program;
3. Availability:
   * Ability to use from a computer with the program installed on it, regardless of the operating system;
4. Reliability:
   * Check for previously entered areas or service points;
   * Logging program errors or warnings;
   * Saving important information entered in the file system and saving the changes made;
5. Recoverability:
   * Recovering a previous session based on changes made to the file system records;

## **Conclusion**

Consideration of this chapter made it possible to demonstrate the importance of developing the software considered in this thesis for the market and for the development of many areas of business and third-party types of human activity. Despite the narrow focus of the program being developed, the potential of its application is great and will allow it to be implemented in many areas.

The availability of alternatives on the market will stimulate a more thorough and high-quality study of the program so that it can compete with the alternatives available in this field. Despite the complexity of the topic under consideration, the program is built with the aim of being as simple and understandable as possible for the end user. This opportunity to make the program easy to integrate into the internal processes of companies and will increase its attractiveness. The considered requirements for the program will make it possible to develop high-quality software that is attractive to the user, competitive and outstanding.

With multiple options available to address the issue posed to this program, there is potential to further improve program performance and introduce new capabilities to achieve better performance and empowerment. Also, the implementation of the developed program for cross-platform use will make this program a universal means of work and focus on improving the program itself with less time spent on adapting the program for different systems.

The developed program can become an excellent solution to the problem of the set thesis, and become a successful project with high-quality implementation and promotion on the market.

# **2. MODELLING**

UML is a modeling language, a general-purpose development language in a software engineering development environment that aims to provide an efficient, understandable and standardized method for describing system design [26].

The task of the UML is to explain the behavior of the system, detect errors and inaccuracies at an early stage of the software life cycle, provide the system design, clarify the software requirements, simplify the software implementation (development) and effectively explain the work of the program to third parties.

To model the system, to show its features, UML diagrams will be used. This chapter will cover the following models [27]:

* Use-case diagram;
* Component diagram;
* Deployment diagram;
* Sequence diagram;
* Class diagram.

These structures will be enough to understand how the program works, separate command-line based version and based on use of graphic user interface. All diagrams represent modelling of the finished prototype.

## **2.1. User interaction models and diagrams**

This subsection will consider the user's interaction with the program, indicating the actions allowed within the program and the program's response to them. To begin with, we will consider the format of user cases, which is the simplest to understand and visualize, which is as abstract as possible from the internal structure of the program. This is followed by a sequence diagram describing in more detail the user actions and indicating the internal structure of the program for processing the received command.

### **2.1.1. Use-case diagram**

To manage service points, the user is given 5 commands: creating a new service point indicating the coordinates of the Cartesian system with automatic finding of geographic coordinates, creating a new service point indicating the geographical coordinates with automatic conversion to the Cartesian coordinate system, editing chosen service point, deleting chosen point, displaying information about chosen point of service. The user can request the optimal service point either for a location specified in a geographic coordinate system or in a Cartesian one.

When the program is launched, a request is made to receive a fragment of the map with certain coordinates of the center (by default, are considered coordinates of the Chisinau center). Then, throughout the entire operation of the program, this map fragment is shown to the user with the points of service placed on it and their optimal service areas. It is possible to switch between the formats for displaying service areas and the points themselves in the form of a Voronoi diagram, or in the form of a map with a diagram plotted on top of it. To work with the map, it is possible to change the coordinates of the center of the displayed region and display the coordinates of the current center of the map.

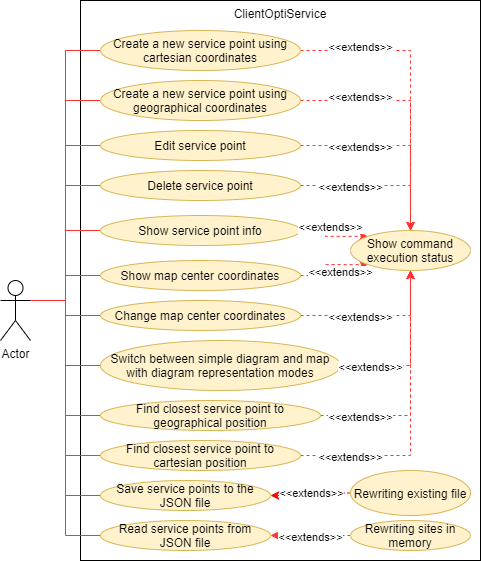


Figure 2.1 - use-case diagram (made by author)

The user is given the opportunity to record information on all points of service in the JSON file with an indication of the path to save the file. If there is a file at the specified path, the program will automatically overwrite it. It is also possible to read all service points from a file. This operation automatically overwrites all points currently in the program memory.

The user has access to operations to determine the optimal service point for a specific location. The location can be specified both in the form of coordinates of the Cartesian system, and in the form of geographic coordinates. When determining the optimal service point, the program will display the found service point on the screen.

The execution of each operation is accompanied by informing in a separate section of the graphical interface about the execution of the command. If there is any error, the user is informed about the requirements for its elimination. All mentioned commands and connections are presented in the figure 2.1 that represents use-case diagram.

### **2.1.2. Sequence diagram**

The sequence diagram has been moved to the appendix section due to its size. Here is described in detail the structure of the sequence diagram and the operation of the program in the format of sequences of events.

The main operations in working with a diagram are creating, editing, showing and deleting sites. When adding a site, the system receives user data and, based on it, creates a new site. When creating, the system automatically finds the missing coordinates for the specified site (if the coordinates of the Cartesian system are entered, then the geographic coordinates are calculated and vice versa). The created site is placed in the site store and the diagram refresh procedure begins. The system takes all available sites and asks for a locus calculation for each. When calculating the locus, the system finds the perpendiculars of all connecting lines with other sites to the current lines and, through operations with them, finds the coverage area. The newly formed locus is assigned to the site, and after obtaining the loci for all sites, an updated Voronoi diagram is created.

Editing a site is similar to the procedure for creating a new site. The only difference is the need to specify the site to be changed, after which the system will search for the specified site. When found, the user enters the same data as when creating and the site is modified. The site is displayed through a site request for display, the system searches for the specified site, and then displays information about it in the system window. The removal procedure is the same as shown, but as a result, the site will be removed from the system and the diagram will be updated.

In the procedures for creating and modifying sites, it is possible to specify both geographic coordinates and Cartesian coordinates. Indicating this aspect within the diagram would take up additional space with minimal differences so it was decided to simplify sequence diagram in this aspect.

When a user requests to write existing sites to a file, the system will ask the file manager to write to a file. If the path is successfully specified, the manager will request all the sites in the storage and write them to a file, after which it will inform the user about the successful operation. When requesting to read from a file, the manager will first check for the existence of the file, and then try to transfer sites from the file to the system storage, overwriting all existing sites at the time of the start of reading.

When changing the coordinates of the map center, the system will automatically recalculate the points available at the time of the change request, redefining the coordinates of the Cartesian system based on the geographic coordinates of the map center and the geographic coordinates of the points. A point is removed from the system if it is outside of the considered area. Based on the new coordinates and the new set of points, the coverage areas are recalculated.

Changing the display mode of the Voronoi diagram to formats with or without a map is performed without recalculating coverage areas for service points. This operation disables the use of a map fragment as a background for a diagram. Coverage areas are recalculated only when factors affecting coverage areas or the location of service points are changed.

When requesting the optimal service point at the specified coordinates, the system will request all sites in the system and check for the presence of the specified location within the locus. As soon as the system finds a suitable locus, it displays the result to the user.

## **2.2. Inner structure models and diagrams**

This chapter examines the internal structure of the program. To begin with, a component diagram will be considered, which allows the program decomposition into a system of interacting modules. The deployment diagram will specify modules location in the system and how they interact with each other. The last step will be a class diagram, which will decompose the entire program into a system of classes that are described in detail by their internal structure and interdependencies.

### **2.2.1. Component diagram**

The current version of the program is based on the definition of a Voronoi diagram using the perpendicular method. Alternative implementations have been cut, but they are all available for review in the legacy source code repository. This clarification will provide a better explanation of the final implementation of the program. First will be reviewed version with graphical user interface, component diagram of which is presented in the figure 2.2.

Due to the peculiarities of the program and the lack of the necessary geometric methods for their implementation in the standard Java graphics library, modules "Line" and "Point" were created to complement their capabilities. To calculate the optimal service area at each iteration and store information about the service area, the "VoronoiPolygon" module was created.

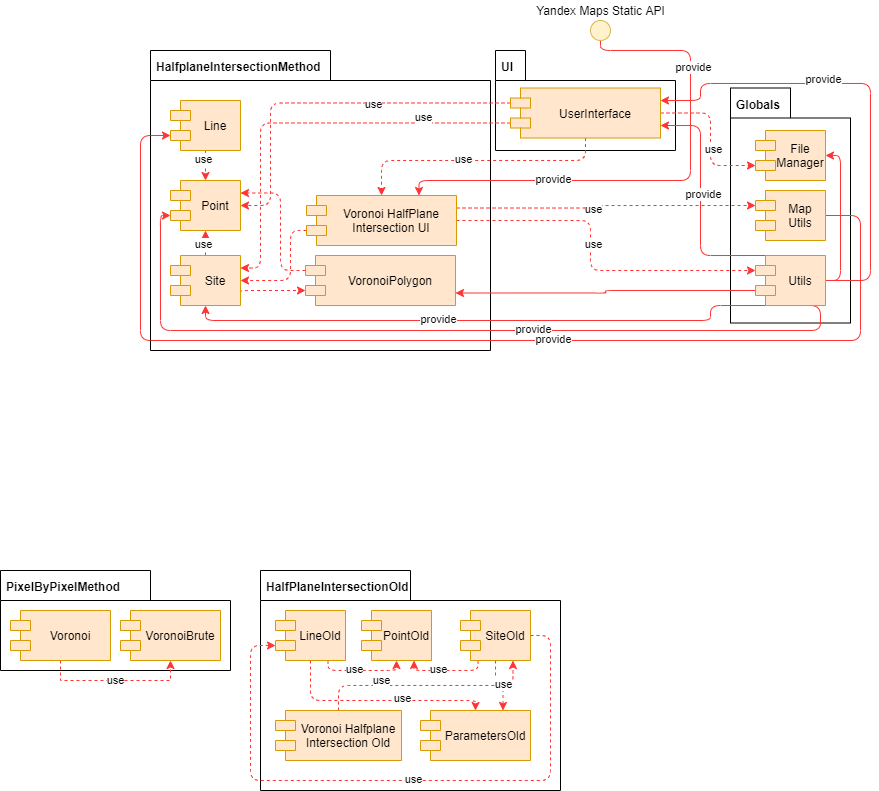


Figure 2.2 - component diagram (made by author)

The "Site" module is the holder of information about the point of service and is responsible for calculating the service area through interaction with all other Sites. For the correct organization of the work of this module and the output of the calculations performed in a graphical form, the "VoronoiHalfPlaneIntersectionUI" module is used. The module additionally executes requests for Yandex cartographic services to obtain a fragment of a map that needs to be demonstrated and is able to request a new one if necessary.

The user interface module is responsible for the graphical interface for working with the program and is a control module that receives user commands and calls the operations corresponding to the command.

Previously, the main modules for working with the program were considered, but it is worth considering the auxiliary ones, which are placed in a separate folder called "Globals". The "Utils" module provides all requesting modules with information to perform operations and acts as a store of service points in memory. "MapUtils" module provides information for working with map services and provides several methods for generating queries or changing them. To interact with the file system, the "FileManager" module was implemented, which provides write / read operations for JSON files.

The general structure of the program implies the mutual use of the main modules for systematization and ease of understanding of the work of the program. The rest provide useful information and small methods to work with.

When looking at the component diagram for a command line program implementation (presented in the figure 2.3), it is noticeable a slight difference from a user interface implementation, but there are a couple of interesting notes.

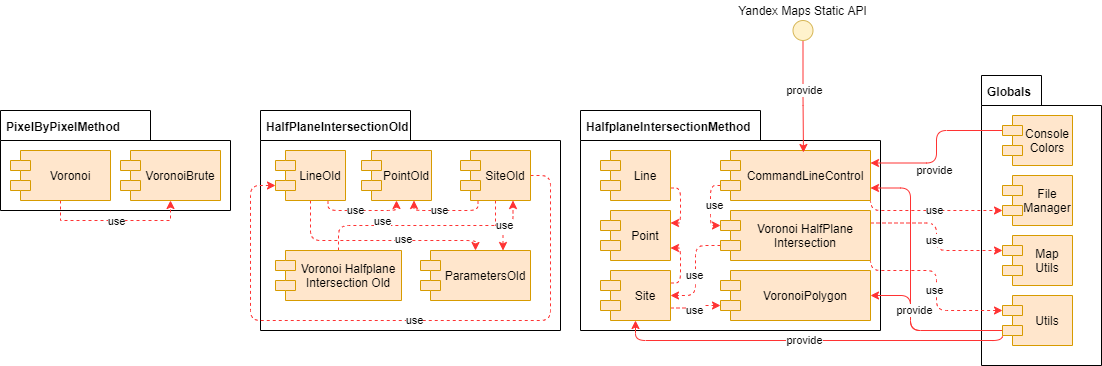


Figure 2.3 - component diagram for command line-based version (made by author)

In contrast to the implementation with a graphical interface, the current one is distinguished by the absence of a graphical interface class and the presence of a class for regulating the color of the displayed text in the command line mode. Otherwise, the structure of the program is identical.

### **2.2.2. Deployment diagram**

The program was built from the start of the development on the principle of the minimum need for an Internet connection and the minimum dependence on third-party solutions. In view of this, it turned out to make the program monolithic and to make the structure of its development as simple as possible, which will greatly simplify the further support of the program.

The program can work independently to build a Voronoi diagram, to work with points, however, the use of maps is critical for this program for the ability to represent the map area for which calculations are made. This is the reason for using Yandex cartographic services to get a map fragment as an image.

Obtaining images is carried out by means of HTTP-GET requests, which are performed only when the program is started or when the coordinates of the center of the displayed area are changed.

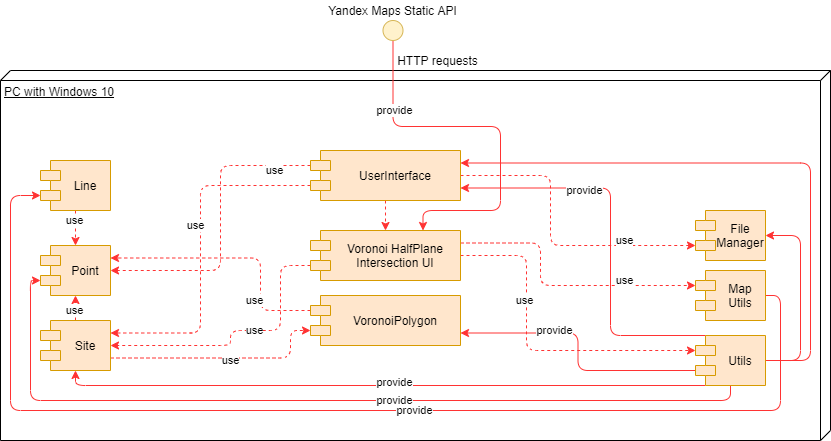


Figure 2.4 - deployment diagram (made by author)

Since the program is written in the Java programming language and does not use system calls or direct calls to the device's file system (all requests are made only by the user), the program can run on any computer that supports Java Runtime Environment or Java Development Kit and requires Internet connection in order to make requests to the Yandex Maps services. Deployment diagram for version with graphical user interface is presented in the figure 2.4.

The deployment diagram presented in the figure 2.5 shows structure for version with command-line based control and it has the same differences as between component diagrams for both versions of the program.

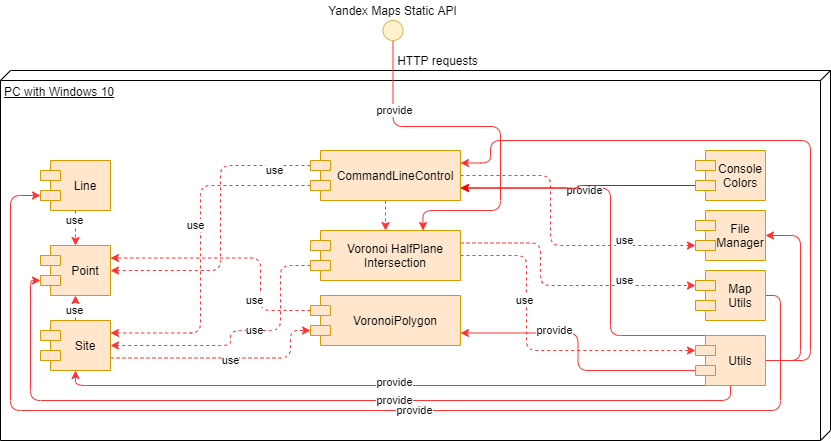


Figure 2.5 - deployment diagram (made by author)

At the beginning of the work, the possibility of implementing the program without using the Internet was considered, however, in this case, the user would have to first download the image of the considered area of the map, which would not allow the program to be self-sufficient and would not provide the possibility of efficient conversion of the coordinates of points.

### **2.2.2. Class diagram**

To begin with, we will consider a version with a graphical user interface, the class diagram of which is presented in figure 2.6. From the class structure of the project, it is worth highlighting the expansion of the capabilities of the original Java classes responsible for working with points, lines, polygons and JLabel, which are libraries of the Swing graphic library (used in Java by default). This was done because the original classes lack the capabilities required for this program or their implementation is inappropriate. Due to these changes, the "Point" class, for example, can now convert coordinates from geographic to Cartesian ones, the "Line" class allows getting a first-degree polynomial that completely describes it and the perpendicular, and the "Polygon" class can "cut off" pieces of its area using a cutting line. Due to the need to draw a Voronoi diagram inside a cell of the program's graphical interface, JLabel class was extended.

The entry point to the program and the controlling class is the "UserInterface" class, which contains all the elements of the graphical interface, listeners for user actions, and calls the required operations. In the final implementation of the program only this class has the ability to interact with the file system through the "FileManager" (which allows read / write to JSON file).

The "VoronoiHalfPlaneIntersectionUI" class is used to draw a Voronoi diagram and attach it to a cell in the user interface. The user interface can work with only one instance of this class and calls it when making any change affecting the drawing of the diagram.

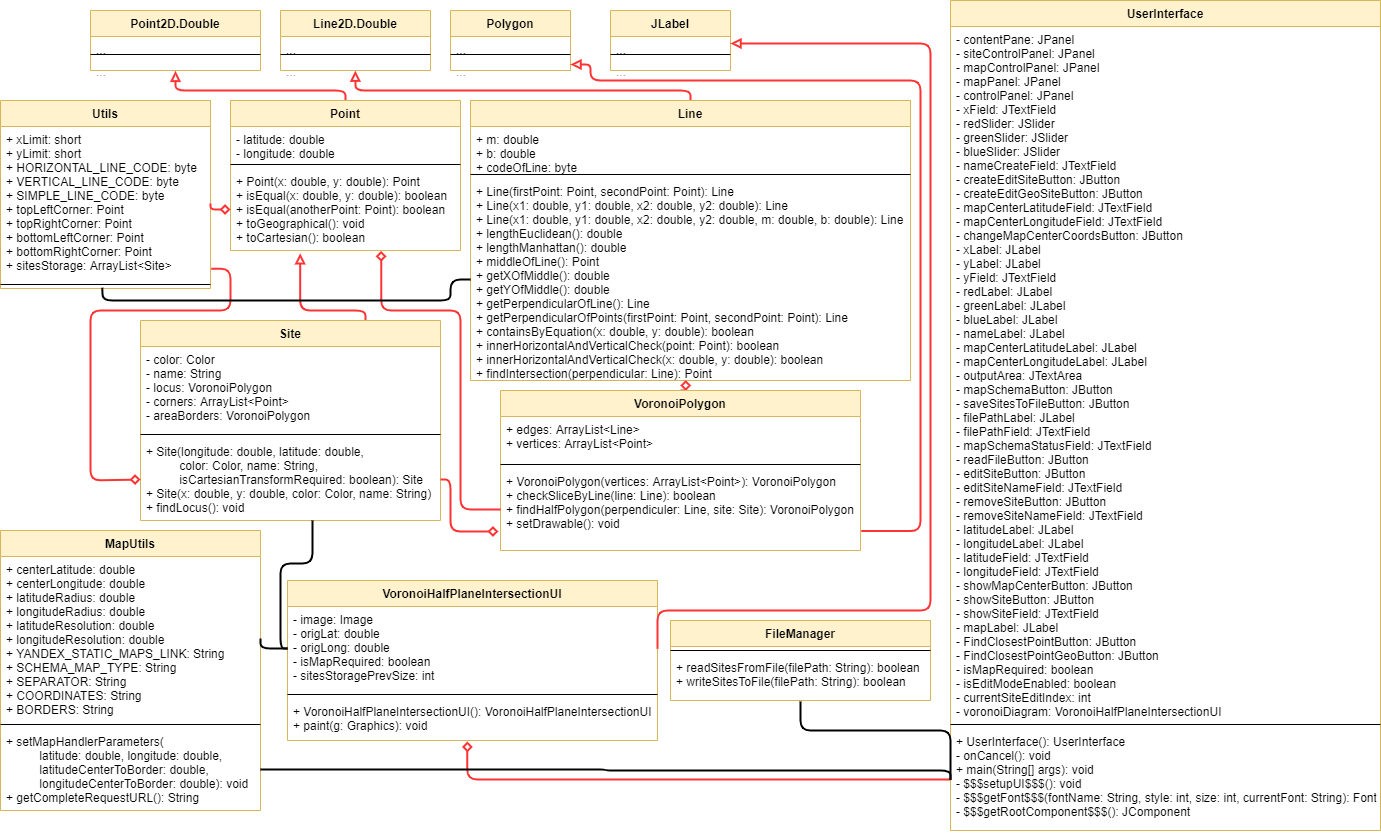


Figure 2.6 - class diagram (made by author)

The basis of the work is the "Site" class, which is the service point under consideration with an embedded locus that can be built using "VoronoiPolygon". Locus building is made via the calling function of searching site locus from the site entity. The locus will cut off areas from itself so as not to fall on other service point coverage areas (the principle will be described in the third chapter).

For the correct operation of geometric operations, the "Point" and "Line" classes are used, which are often used in the "Site" and "VoronoiPolygon" classes. To obtain general information by the "Site", "Point", "Line", "VoronoiPolygon" classes, the "Utils" class is used, which consists of globally available data fields.

The "MapUtils" class is associated only with the "VoronoiHalfPlaneIntersectionUI" and "Point" classes, since for the first it provides functions for calling a map fragment from the map services and changing the coordinates of the required area, and for the second class it provides information about the current coordinates of the map center for converting coordinates from a Cartesian system to a geographic one and vice versa.

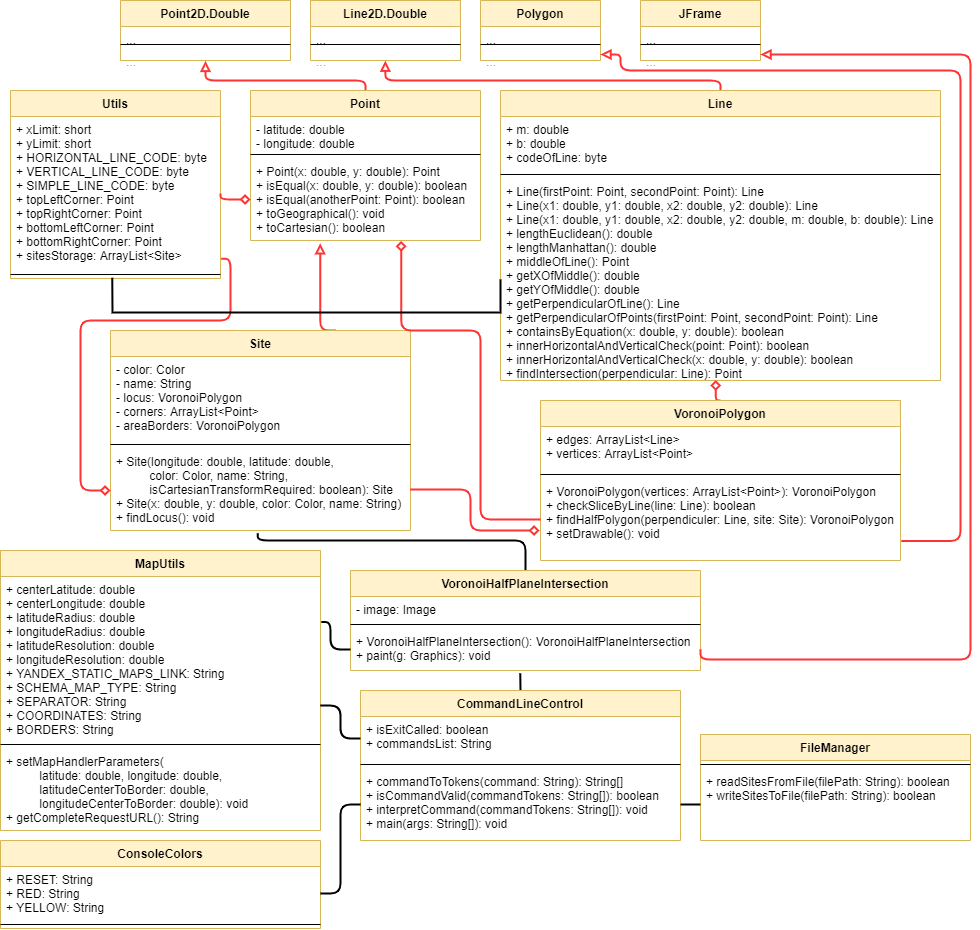


Figure 2.7 - class diagram of command line-based version

The class structure of the command line version presented in the figure 2.7 shows minimal differences from the graphic user interface version. The difference is the use of the command line class as the entry point, which uses the command line text color class to display colors correctly, and the class for finding the Voronoi diagram inherits another class, since it will be displayed as a separate window when prompted.

## **Conclusion**

This chapter has demonstrated several important aspects of how the program works.

Based on the diagrams of user interaction, the simplicity of working with the program and its construction is noticeable to minimize the actions required from the user. Such a structure of work minimizes the threshold of entry for working with the program, allowing it to be used for work by almost any person.

Diagrams of the internal structure of the program show the solidity of its implementation and dependence only for obtaining a fragment of the map, since all other operations by the program are performed independently and locally. Despite the solidity, the internal structure of the program is quite complex and confusing, due to the complexity of the tools implemented for the work. The system uses many types of relationships between classes and complements the capabilities of the standard Java graphics libraries for full-fledged work, which also led to the complication of the internal structure of the program.

The use of third-party libraries and solutions would simplify the class structure of the program, but the program would then become more dependent on third-party solutions and would significantly increase the amount of both resources spent and the volume of the program. It is worth noting that with this approach, updating the program when a new version of Java appears would become difficult due to the dependence on outdated libraries and third-party modules.

The structure of the program allows fairly simple unloading of it as a Docker image and use as a microservice if necessary. It is worth noting that the implementation with the command line allows using the program without rendering the interface, which is even more optimized in terms of resources, but more difficult in terms of user control. The provision of diagrams of the implementation of two versions of the implemented software at once is necessary for a complete understanding of the processes of operation of both versions, visualization of their differences, and demonstration of the proximity of these approaches.

Based on the diagrams shown and the described concepts of work, it is possible to review the progress of the program development with consideration of the most significant development stages.

# **3. DEVELOPMENT**

The previous chapter demonstrated the structure of the program, its presentation in the form of various diagrams, diagrams and other structures. In view of the consideration in the last chapter of the class diagram, which clearly demonstrates the structure of classes and their methods with fields. For this reason, the current chapter will only consider the implementation of the algorithms for defining the Voronoi diagram in the form of code at the moment.

## **3.1. Algorithms for constructing a Voronoi diagram**

In this subchapter, the implemented methods of constructing a Voronoi diagram will be considered. Each method is accompanied by a description of the mathematical implementation of each method with diagrams that visualize the principles by which the program works.

### **3.1.1. Pixel-by-Pixel method**

The pixel-by-pixel method for determining the Voronoi diagram was the first working prototype, based on which subsequent implementations were obtained. The part of the source code corresponding to this method was cut from the program, but is available at the link to the repository of the legacy version of the program. In annexes nr. 2 and nr. 3, the detailed commented-out source code of the constructor of this method and the main part of the algorithm is attached.

The calculation of the Voronoi diagram using this method is based on the coloring pixel in the color of the nearest site. Determination of the distance from the site to the pixel is carried out by one of two methods of determining the distance in the Cartesian coordinate system: the Euclidean method or the Manhattan method [28]. The first one determines the distance according to the formula (this one was chosen as default) [29]:

(3.1)

The second method uses the formula [30]:

(3.2)

This implementation gives an accurate image of the Voronoi diagram, but does not write the coverage areas of the service points in memory, since the faces of the polygons that describe them are not calculated. It is possible to write an algorithm for converting a colored area into a polygon through a pass through all pixels of the specified area, but this will make the algorithm, which is already complex in execution time, even more complicated.

Any change to the site will entail a rebuild of the entire diagram, which reduces the efficiency of the algorithm. It is also important to note that the algorithm for determining the distance by the Euclidean method is more complicated than the Manhattan method, since it requires the execution of exponentiation operations and finding the square root, in contrast to the formula based on addition and subtraction, indicating the absolute values ​​of numbers.

Building a Voronoi diagram for 100 sites using the Euclidean formula will give the picture presented in the figure 3.1.

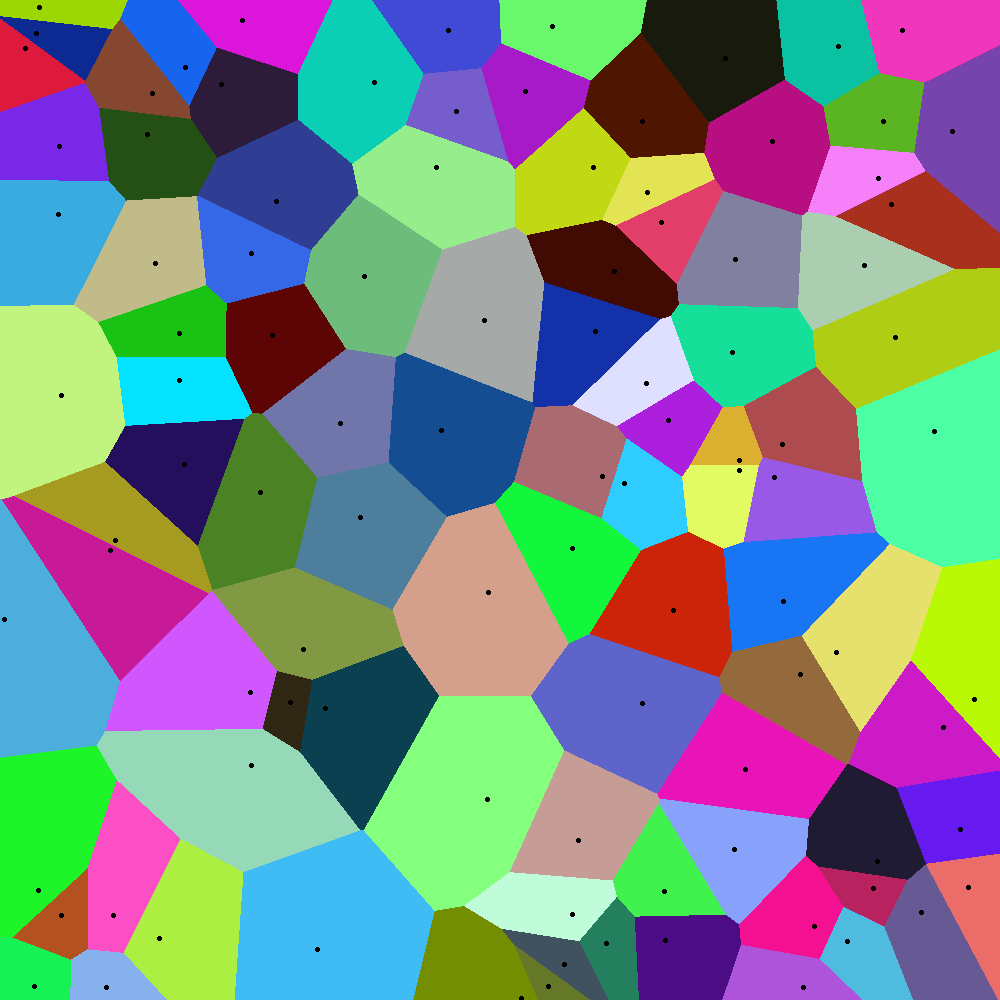


Figure 3.1 - result of launching per-pixel method (made by author)

As it can be seen, the coverage areas are constructed optimally, however, plotting a diagram for 100 sites using the Manhattan method will give a different result presented in the figure 3.2.

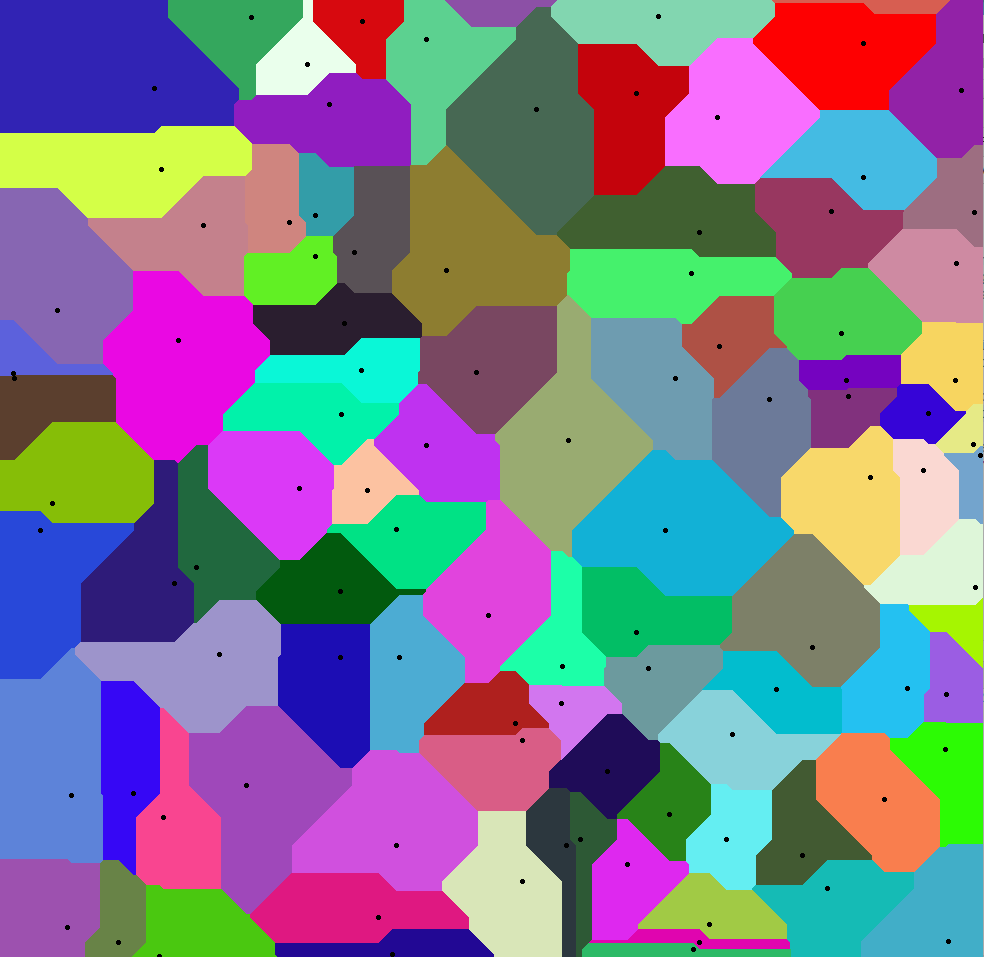


Figure 3.2 - result of launching per-pixel method with Manhattan algorithm (made by author)

The first diagram visually better matches the actual coverage area of the service points, but it should be noted that the Manhattan method also allocates coverage areas quite efficiently. The previously considered time complexity still does not allow using this method as the main one and it is necessary to implement a more efficient method for determining the Voronoi diagram.

### **3.1.2. Perpendicular method with half-planes intersection**

In the first chapter, the principle of constructing a diagram by the method of perpendiculars was considered, but now it is worth delving into the mathematical approach to describing this method.

A straight line can be constructed between two points in the Cartesian coordinate system, which is described by a polynomial of the first degree according to the next formula [31].

(3.3)

To simplify, the constant b is responsible for the displacement of the straight line relative to the X-axis and the constant m is responsible for the angle of inclination of the straight line with respect to the X-axis. When m is equal to 0, the straight line is parallel to the X-axis, with a value above zero, the straight line is turned counterclockwise relative to the X-axis, and if the value is below zero - clockwise relative to the X-axis.

The coefficient m is found by the formula:

(3.4)

Using one of the points coordinates can be found the coefficient b:

(3.5)

Knowing the polynomial describing the straight line passing through these points, can be found the perpendicular of this line. Perpendicular coefficient m can be found using formula:

(3.6)

After finding m coefficient can be calculated the coefficient b for the perpendicular so that it will cross the line in the middle:

(3.7)

Middle of the segment can be found using next formula:

(3.8)

Code responsible for performing perpendicular calculation between two points is presented in the annex nr. 4 (also available in the source code repository)

After carrying out the above calculations, the program obtains the coefficients m and b of the first-degree polynomial, which fully corresponds to the given perpendicular, and a line is built bounded by the considered area. The rest of the straight line is ignored.

Due to the limitation of the line within the considered area and when considering the area as a plane, the division into two half-planes by this line will be obtained. For further work, it is required to build a polygon that will correspond to the half-plane containing the site. How to do it? For better understanding it is necessary to review figure 3.3.

It is necessary to find the half-plane that contains the service point *S2*. The program starts moving from the upper left point (point *A*) and moves in a clockwise direction. Upon reaching the first point of the perpendicular, the algorithm fixes the first found vertex of the half-plane, and all further encountered angles will be fixed as vertices of the half-plane.

Upon reaching the second point of the perpendicular, the algorithm fixes the last point of the half-plane and forms the half-plane. In this case, the half-plane is obtained. The algorithm then checks to see if the service point is inside the half-plane. Upon confirmation, the program returns the found half-plane, otherwise the second half-plane is taken.

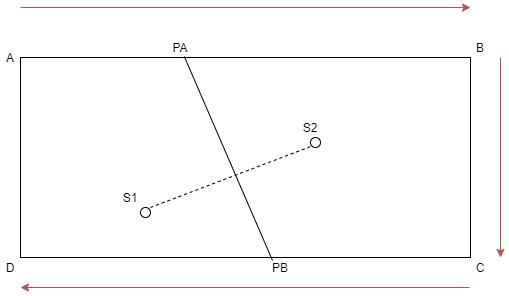


Figure 3.3: method of constructing half-plane (made by author)

Code responsible for performing this algorithm of finding halfplane is presented in the annex nr. 5 (also available in the legacy code repository).

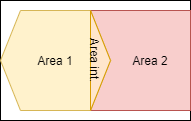


Figure 3.4 - intersection of two areas (made by author)

Due to polygon limitations in Java, it was decided to keep the half-plane as an area for area intersection operations, for which the Area class was used which works conform presented in figure 3.4 principle.

The algorithm consists in traversing all the lines forming the figures for which the intersection is calculated. The nesting of lines of one shape into another is determined, their intersections are checked, and based on the data obtained, the intersection of the regions is determined. Lines describing an object of the Area class are stored considering that they may not be straight lines and therefore this method excludes the possibility of extracting the vertices of the regions (in this situation they can only be polygons) or their faces. The complexity of this calculation is high due to the complexity of the checks carried out during the calculations and the format of the calculations that is not optimal for this task. An additional problem is the need to keep in memory geometric lines describing areas. In addition, such a structure of the program can lead to its unstable work in the presence of a large number of sites. The program memory will be full and the program may terminate abnormally.

Running this algorithm will display the Voronoi diagram for 100 points shown in the figure 3.5.

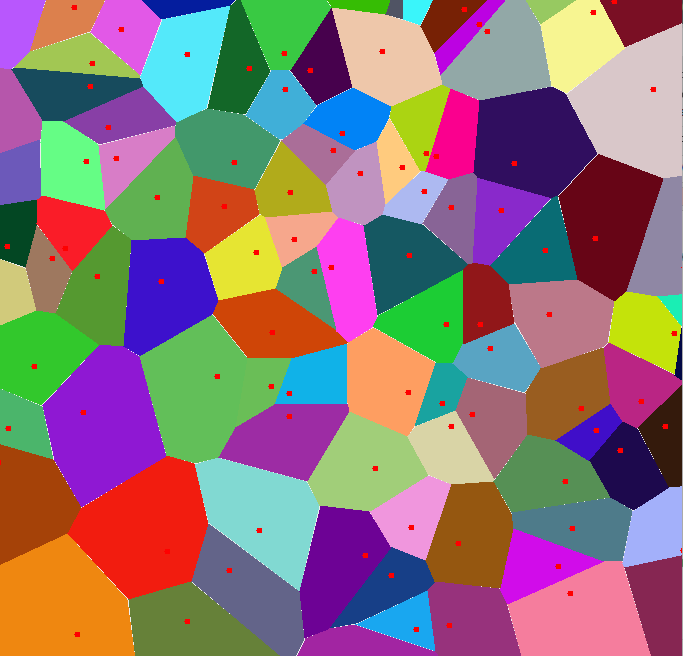


Figure 3.5 - Voronoi Diagram made by perpendicular area method (made by author)

Despite the high-quality implementation of this method, locus detection by intersecting regions is not an optimal solution, and the lack of an opportunity to obtain the faces of the polygon, which is the site locus, makes the method controversial and difficult to use.

### **3.1.3. Perpendicular method with polygon clipping**

To improve the mentioned before algorithm, it was necessary to correct the principle of calculating the intersection of half-planes. When considering the method for finding the perpendicular and the principle of the intersection of objects of the Area class, the idea of cutting off the polygon sector by perpendicular appeared presented in the figure 3.6.

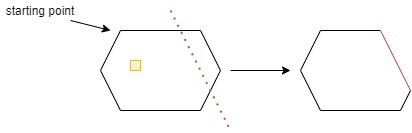


Figure 3.6 - algorithm for slicing polygon by perpendicular (made by author)

The polygon has its own entry point, from which the path begins along the edges in a clockwise direction. When an intersection of a face with a perpendicular is detected, the beginning of the half-plane is fixed and a potential entry point is fixed. Each next vertex is written into a new half-plane, and as soon as one more point of intersection is reached, it is fixed as the final point of the new half-plane. Then the half-plane is checked for the presence of a site in it. If the condition is met, then this half-plane (polygon) is written in memory as a new site locus. Otherwise, from the second intersection point, the program iterates further to the first intersection point, writing all the vertices encountered in a new half-plane that will contain the site.

The advantage of this algorithm is that the complexity of locus determination is significantly reduced, since in the worst case, the algorithm will walk over all polygon faces. This is much more efficient than checking the boundaries of two shapes, checking their nesting, and other factors. An auxiliary factor that improves the attractiveness of the algorithm is the ability to query the vertices of the resulting polygon, its faces, which improves the applicability of the program.

To avoid the possibility when the algorithm will try to cut the polygon if the perpendicular does not cross it, before calling this operation the program checks the condition: if the line intersects the polygon twice, it means that the perpendicular "cuts" part of the polygon. This operation is a part of the Polygon class extended for this task.

Code responsible for performing slicing mechanism of finding site locus is attached in the annex nr. 6 (also available in the source code repository)

## **3.2. Additional features**

The current chapter contains information on the topic of additional implemented functions that expand the possibilities in working with the program.

### **3.2.1. Yandex Maps Static API**

To demonstrate the real coverage areas of service points, it was decided to use third-party programs to obtain map fragments. Services for generating a static map image based on HTTP requests are suitable for solving this issue, but it is worth mentioning a few important notes.

Static image generation services provide their services in a limited format. Google maps, for example, provides an opportunity to make free 25 000 requests per day, but these questions are assigned to a unique usage key, and to generate a key, user must register on the Google Maps platform. Therefore, either each unit of the program must have its own unique key, which will require an additional step of registration on the specified platform from the user, or it will be necessary to purchase a paid tariff for using cards, which may be ineffective for the purpose of creating a working prototype of the program.

Such a binding of the service to a unique key is common and is observed on many analyzed services, limiting effective use of the program. The only found by the author service that provides its services without binding to a key is the Yandex Maps service.

|  |
| --- |
| /\*\*  \* Form string-formatted URL that will be used for performing request  \* **@return** string-formatted URL for making request to the Yandex.Maps Static API  \*/  **public static** String getCompleteRequestURL() {  **return** YANDEX\_STATIC\_MAPS\_LINK +  COORDINATES + centerLongitude + "," + centerLatitude + // center of the map coordinates  SEPARATOR +  BORDERS + longitudeRadius + "," + latitudeRadius + // radius of coverage by map  SEPARATOR +  SCHEMA\_MAP\_TYPE;  } |

Listing 3.1 - principle of getting request URL to the Yandex Maps

Yandex Maps allows making 1000 requests per day to receive static map fragments. This amount is enough for checking and testing the program's work and will allow several units of the program to work independently as needed. The service is based on obtaining a picture of the desired fragment of the map by sending a GET request along a certain path, indicating the parameters of the required picture. The principle of generating a request is shown below.

The program stores information about the path, the types of arguments used, and the argument separator. The program receives the coordinates of the center of the area under consideration and the size of the area under consideration in degrees of longitude and latitude. Based on all the above parameters, an URL is generated, by sending a request for which the program will receive an image of the desired fragment of the map, however, it is important to make a clarification here.

By default, Yandex Maps returns an image with a resolution of 600 and 450 pixels. Since the size of the requested area may not fit into the generated image entirely, the service resorts to one of the "correction" methods - the map slightly enlarges the coverage area so that the requested area is inside this area, or the map cuts off the area along the axis that does not fit into the specified image resolution. In order to perform point transformation from Cartesian coordinates to geographical ones and vice versa. The program takes length in longitude and latitude, then takes size of image in pixels and calculates length in longitude and latitude for each pixel. Using an estimated coefficient program easily finds all required coordinates.



Figure 3.7: principle of cutting fragment by Yandex Maps Static API, where red rectangle shows actual received map fragments (made by author)

When specifying an area on the map, the correct image of which the cartographic service cannot provide, the image is adapted to this area either by cropping the image or by reducing the zoom factor of the image (this problem is presented in the figure 3.7). Thus, to increase the stability of this method, the algorithm will only request an area along a certain radius.

### **3.2.2. JSON sites storage**

Since the number of sites can be quite large, the ability to save sites inside a JSON file was added. The choice of this extension is due to the desire to provide the ability to transfer the results of the program to other applications, which is convenient to implement by importing data from files of this format.

In the figure 3.8 is presented on the left the structure of the site storage file, which is made in an easy-to-read style for the user, even without the use of third-party programs. The use of JSON files allows in the future to transfer information to third-party programs and, with further improvements of the program, not to lose the information entered by the user.

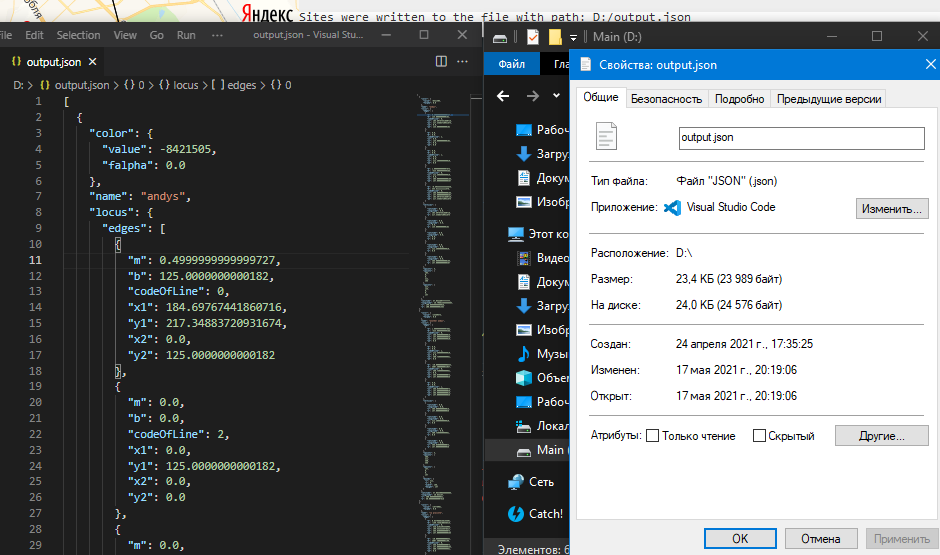


Figure 3.8 - JSON file storage structure and properties of this storage (made by author)

The program allows saving data in any path specified by the user and also read from any file. When an attempt is made to write to an existing file, the program will overwrite its contents, which makes it possible to update the information stored in the file as needed.

### **3.2.3. Graphic User Interface**

To work with the program, a graphical interface was implemented, which is presented in the figure 3.9 and is divided into three areas. The area on the right represents the user interaction area. The first half of the area is devoted to the basic operations of working with the program, namely, creating a new site based on the coordinates of the Cartesian system or geographic system, color and name parameters, as well as finding the optimal service point for a specific location indicated by the coordinates of either the Cartesian system or the geographic system.

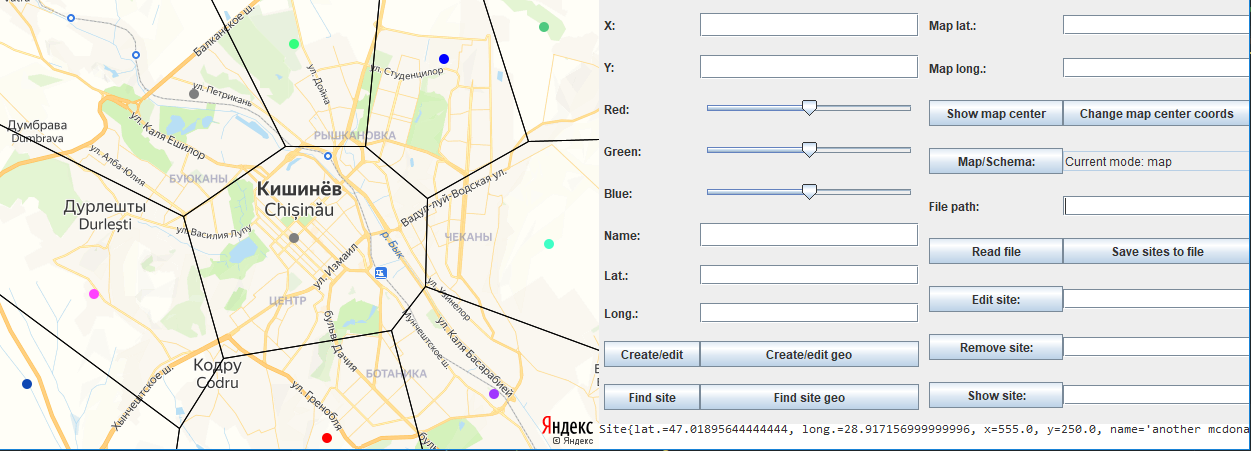


Figure 3.9 - final program build with graphical user interface (made by author)

The second part of the program is located under the control panel and is used to provide the user with the results of the execution of non-graphical operations by the program and inform the user, if there are errors, about the methods of their elimination. The panel is non-interactive (allowing only copying of the displayed text), which excludes the possibility of deleting useful information by the user.

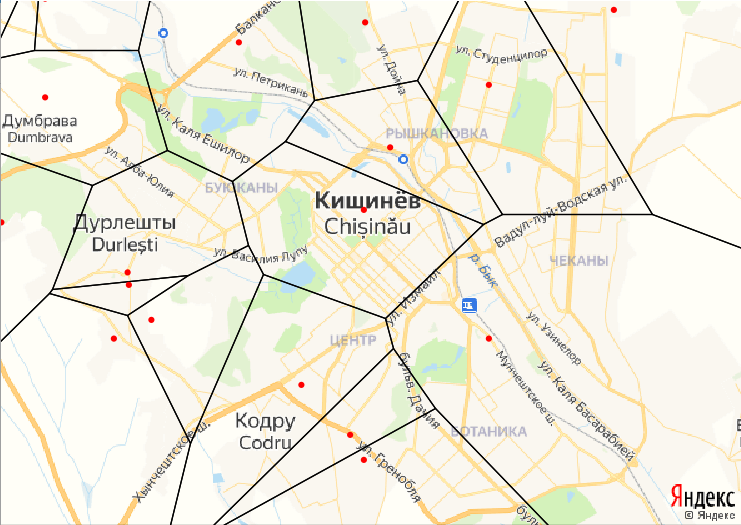


Figure 3.10 - Voronoi diagram shown for 20 sites on the map represented by red dots and their optimal service areas (made by author)

The third part of the program is located to the left of the control panel and is used to visualize the resulting Voronoi diagram applied over the map or without it (this part is shown in the figure 3.10). When executing each operation affecting the coverage areas of the service points, the display of the sites themselves, the display parameters of the Voronoi diagram, the map visualization parameters, the panel automatically changes the visualization of the specified area, applying the changes made. This structure allows the user to immediately see the results of their actions. Due to the previously performed actions to optimize this algorithm, each update of the diagram is performed quickly and with minimal expenditure on the resources of the computing device. In addition, when recalculating visualization, the program will recalculate loci (coverage areas) only if there are changes in service points.

When developing the user interface, the emphasis was placed on the most efficient organization of the space available to the user, while minimizing the involved elements of program operation control. The design of the program requires further development in the future to obtain a more user-friendly appearance of the program. In addition, the design was built with an emphasis on the minimum requirements for the graphics module of the computer.

Despite the emphasis on interface efficiency and less on user-friendliness, the displayed diagram and map area have been redesigned to improve image quality and clarity using anti-aliasing technologies to better visualize results. These actions were also performed using the Java language tools available within the default tools. Due to the limitations of cartographic services, the resolution of the resulting image is strictly limited, however, the program is built in such a way that with the further implementation of other resolutions of the resulting fragment of the map, the source code of the program does not need to be processed.

If there is a need to improve the resolution of the resulting map fragment, user will have to use paid licenses for working with Yandex Maps, which will affect the cost of the program.

## **3.3. Performance test**

During the development, much attention was paid to building an algorithm that is efficient in terms of performance. To consider the performance progression as the program develops, three previously described Voronoi diagram solutions will be considered: the per-pixel solution, the intersecting half-plane perpendicular method, and the polygon cut-off method.

Table 3.1 - Comparison of single thread performance for used computers (made by author)

|  |  |  |  |
| --- | --- | --- | --- |
| Name of used CPU | AMD Ryzen 5 4500U | AMD Ryzen 5 3600 | Intel Core i7-7700HQ |
| Single thread performance (MOps/s) | 2,472 | 2,584 | 2,07 |

The table above shows the processors of the computers used for performance tests. In view of the implementation of the program for operation in the single thread mode, the indicators for the corresponding operating mode of the processors are indicated [32] [33] [34]. For efficiency of the analysis, the launch indicators for 100, 1000 and 10,000 service points will be demonstrated, which will allow checking the operation of the program under load. Performance was measured for the site loci determination operation, which is the basis of the task specified in the work.

Table 3.2 - Voronoi diagram calculation time, pixel-by-pixel method (made by author)

|  |  |  |  |
| --- | --- | --- | --- |
| Pixel-by-pixel method (ms) | 100 service points | 1000 service points | 10000 service points |
| AMD Ryzen 5 4500U | 817 | 7131 | 50672 |
| AMD Ryzen 5 3600 | 521 | 5084 | 50292 |
| Intel Core i7-7700HQ | 753 | 6782 | 65139 |

The execution time of the task by the pixel-by-pixel algorithm is unsatisfactory and the use of such an algorithm for determining the loci will be suboptimal. The increase in the complexity of the execution time of the algorithm is high and in the presence of a small number of sites under consideration could be used, however, such performance indicators would adversely affect the applicability of the method by users.

Table 3.3 - Voronoi diagram calculation time, perpendicular area intersection method (made by author)

|  |  |  |  |
| --- | --- | --- | --- |
| Perpendicular with Area intersection method (ms) | 100 service points | 1000 service points | 10000 service points |
| AMD Ryzen 5 4500U | 179 | 4162 | 322212 |
| AMD Ryzen 5 3600 | 140 | 3836 | 336390 |
| Intel Core i7-7700HQ | 219 | 4965 | 318292 |

This case is more interesting to consider. The dynamics of the search for site loci with their small number has become several times better compared to the pixel-by-pixel method, leading to a reduction in the computation time for 100 points by about 4 times. Despite such a favorable picture, already at the stage of calculations for 1000 points, the performance gain is no more than 40-50%. When calculating a large number of points in the amount of 10,000 sites, the calculations go beyond the acceptable time frame and are significantly inferior to the efficiency of the per-pixel algorithm.

Such a negative picture of the performance progression as the number of points grows is tied to the time complexity of the algorithm established earlier. However, judging by the indicators, the algorithm works in a suboptimal way, which leads to such indicators. Earlier, the main limitation of this method was established - the use of the technique of intersecting regions was performed in the standard Java library in a non-optimal way, which made the implementation of this operation require improvement.

Table 3.4 - Voronoi diagram calculation time, perpendicular cut-off polygon method (made by author)

|  |  |  |  |
| --- | --- | --- | --- |
| Perpendicular with cut-off polygon method (ms) | 100 service points | 1000 service points | 10000 service points |
| AMD Ryzen 5 4500U | 15 | 185 | 7460 |
| AMD Ryzen 5 3600 | 17 | 135 | 5258 |
| Intel Core i7-7700HQ | 32 | 322 | 7826 |

The performance of this method is superior to both previously presented methods. For a small number of sites, the performance gain relative to the pixel-by-pixel method is about 30 times, and compared to the intersecting regions method, about 8 times. The time indicators are extremely small, which indicates the improvement of the algorithm to an acceptable state. When considering the average number of points, the execution time compared to the per-pixel method was reduced by 37 times, and compared to the area intersection method by about 28 times. The difference in execution time for a large number of points is especially noticeable, since the search time for a large number of points in comparison with the per-pixel algorithm has been reduced by about 6-7 times, and compared with the method of intersecting regions it has been reduced by 40-41 times.

The performance of the last method achieves results that can be effectively used within limits of the estimated task and used as the main one. Thanks to this algorithm, the problem can be solved in a short time with minimal use of computing resources.

### **3.3.1. Opportunities to improve performance**

There are several methods for further improving the speed and efficiency of the algorithms. Here they will be discussed.

The principle of determining the site locus is implemented in such a way that the calculation is performed independently and depends on this calculation on the coordinates of other sites. Due to the peculiarities of the algorithms, the number of these sites does not change during computations, which makes it possible to distribute the computation of site loci by parallelizing computations.

Parallelization of computations can be an effective solution to the performance problem posed during work over the algorithms, but it is worth remembering the limitations of Amdahl's law in order not to achieve a performance drop due to constant switching between threads. Parallelization has not been implemented for minimum computational resource requirements.

An alternative solution to improve performance is to use algorithms with better time complexity, such as Fortune's algorithm or the recursive algorithm. The first chapter examined the complexities of these implementations and the reasons for refusal in the current work from them. However, due to the better time complexity, their implementation would have a beneficial effect on a large number of analyzed sites.

An auxiliary method for improving performance could be the use of third-party graphics libraries, which have a wider set of geometric capabilities and operations, but their use will adversely affect the resource requirements of the program, make the program dependent on third-party modules, limiting the possibility of software updates and the need to master new building tools Voronoi diagrams.

## **Conclusion**

The implementation of the program based on the principles outlined in the first chapter and on the basis of the models and concepts discussed in the second chapter became a rather laborious and complex process that required a thorough approach to writing code. Due to the complexity of the topics and algorithms under consideration, throughout all stages of writing the code, all actions were actively documented and commented within the code of the algorithms and tasks being implemented.

Based on the concepts from the first chapter, it was possible to build a program that is easy for the user to learn, which reduces the entry threshold for using this program to a minimum. Since the implementation was important for the program in such a way that it could be used both as a simple user application and as a program for working through the command line, which makes it possible to work with it as a microservice with further refinement or to maximize resource savings from - due to the absence of resource costs for rendering the user interface. The program was built according to the requirements of the first chapter, which allows us to consider the program as a competitive solution.

Based on the models and structures from the second chapter, it was possible to build the program in a systematic and convenient way for the possibility of its further improvement. Despite the implementation of the two versions of the program, their versions are extremely similar, which minimizes the costs and difficulties for transferring changes from one version to another. Due to the simplicity of its structure, it can be considered as a complete solution that can be implemented in third-party platforms and systems.

By analyzing the implementation of solutions to the problem indicated in the thesis, it was possible to select effective algorithms for the program. Due to constant improvements to the code and algorithms, it was possible to build a program with acceptable performance indicators and minimize the resources spent. Despite the amount of work done in this direction, it is possible to refine the program by using more efficient algorithms for the program and implementing the parallelization of the program. Due to the complexity and limitations of these aspects, it is recommended to further refine and improve the program through the efforts of a small team.

Due to the limitations of the data used and the peculiarities of plotting the diagram, the program contains minor limitations and errors. To maximize the efficiency of the program and the further possibility of implementing new tools, it is recommended to introduce current technologies into cartographic services or gain access to more advanced technologies for working with maps, such as vector maps with complex sets of information. Despite the advantage of this improvement, the program will expand and become more complex, negatively affecting the user experience and ease of use.

The obtained implementation can be used to eliminate the problem of the thesis and can be provided to the end user for work; however, it is recommended using the resulting program to further improve it and bring it to a more stable, complex and easy-to-use form. This work can become a qualitative basis for building a complex cartographic service.

At this stage, the consideration of the theoretical basis of the thesis is completed, the consideration of the work as a scientific project, and then the implementation of the project as a startup with an analysis of the economic component, determination of the prospects of the project as a commercial product will be considered.

# **4. ECONOMIC RESEARCH**

Based on the developed prototype, the work done on modeling the program, considering the situation in the area under consideration, it is possible to draw up an approximate position of the program as a product to be realized. During this chapter, the factors that determine the position of a product on the market, its possible impact on the market in general and on buyers in particular, as well as other economic factors will be considered.

## **4.1. Business case**

The task assigned to this project is to create an effective tool for determining the coverage areas of service points with the possibility of easy management of the objects in question, changing the considered area, for visualizing data in the form of cartographic diagrams and classical schemes applicable for cartographic and mathematical analysis, with the ability to determine the optimal service point based on nesting in the service area of ​​the location in question. Within the structure of the project, the implementation of the product is required to be able to be used both as an easy-to-use graphical tool for an ordinary user, and as an acceptable one for implementation in more complex cartographic services or third-party solutions of a microservice operating on a command basis. Such a variety of available solutions will expand the coverage of the user base, the applicability of the solution.

Below is a table with general information on business case analysis.

Table 4.1 - Business case of the project (made by author)

|  |  |
| --- | --- |
| **Project:** ClientOptService | **Version:** 1.0 |
| **Executive summary** | Service Point Coverage Service to further determine the optimal service point for the area of interest and map analysis. |
| **Reasons for this project** | * Lack of alternatives in the market * High cost of alternative solutions * High complexity of work with alternative solutions * Innovation in the context of easily accessible map analysis tools to simplify analytics and optimize processes within companies * Desire to create a quality, easily accessible and easily applicable product |
| **Option 1** | Using the solutions provided by Google maps and Yandex maps to find the shortest route in time or distance, followed by manual analysis of the optimal service area |
| **Option 2** | Fully manual calculation of the optimal service area with recalculation of the area when changing |
| **Option 3** | Mastering the ArcGIS cartographic analysis tools to determine the coverage areas of service points |
| **Timescale** | Working prototype in 1 month, fully working application in 1.5 months |
| **Major Risks** | * The emergence of an alternative from a large company that can integrate the solution into a large service like Google Maps * The emergence of an alternative solution that will have a similar set of functions, but will be aggressively advertised and less expensive * Potential disinterest of corporate clients in the program * A significant reduction in the cost of the ArcGIS toolkit and its breakdown into simpler and narrowly focused sets of tools. |

For this project, it is also worth defining its business objectives:

* Creation of an algorithm for finding a coverage area by a service point for cartographic analysis and for allocating resources between service points;
* Calculation of the covered areas in real time with any change in service points;
* Structuring the program for the possibility of use by individuals, legal entities and for implementation in third-party applications;
* Minimization of the required computing resources to be able to use the program on most computers;
* Creation of an easy-to-use and easy-to-learn tool of work.

In view of the requirements for the program and the implementation of an application that requires the development of effective algorithms to develop an effective solution and obtain a competitive product, it is worth using a small development team, which will be responsible for the detailed and high-quality implementation of the product. Development is also possible by one developer (which was done to develop the resulting prototype), however, in order for the program to comply with all the requirements, it will be necessary either to abandon some additional program features, or to increase the project development time, which will be discussed below. Recommended case is to use a small development team, activity planification and management of which will be discussed further in this chapter.

## **4.2. Target audience/market segment**

The deployment of the application only for the local market can bring significant income, since the total number of legal entities registered in the Republic of Moldova is about 233 thousand [35] and companies providing goods / services delivery will be interested in using the application. Even if we consider that only every tenth company may be interested in our program, the number of potential clients is about 23 thousand legal entities. Such companies are becoming more and more due to the massive adaptation to the online format of interaction with customers due to the Covid-19 pandemic. Only 4 months after the start of the pandemic in the Republic of Moldova, the number of enterprises that expanded their online activities increased by a third, introduced delivery by a third, and started online activities by a quarter [36].

The current situation in the private business market increases the need to use algorithms, techniques, programs to optimize delivery processes, select the optimal service point, considering all factors. Medical institutions, ambulance services, police, firefighters and other government agencies will also be interested in using such software to improve service.

Expansion of the client base is possible by promoting the program among foreign clients. Private companies in other countries are also actively moving to the online model with the delivery of goods/services [37] [38], therefore, there will be demand from other countries as well.

In the United States alone, there were about 500 thousand companies for the delivery of goods, 970 thousand companies for the transportation of goods [39] [40]. There are 12 to 24 million e-commerce platforms recorded worldwide [41]. Even if it is considered that only every fourth platform provides delivery of goods / services, then there are up to 4-6 million electronic distribution platforms that can become users of this program. Finding statistical data on the number of companies for cargo transportation and passenger transportation or companies providing delivery of their goods / services is not possible, but based on these figures, it is noticeable the large number of potential client bases.

It is important to note the applicability of the program for the IT-industry, since the program can optimize the load on data centers, correctly distribute user requests across servers depending on location, and so on. There are about 8.5 million data centers registered in the world [42], about 70 thousand providers are registered [43].

The specificity of the developed program determines the possibility of its application for logistics companies, since it can favorably affect the ease and speed of cartographic analysis and determination of delivery channels. Given the size of these companies, the number of divisions of companies in different countries and the amount of income received, there is the possibility of profitable sale of goods with its effective positioning and bringing it to a high level of quality [44].

Government services and enterprises, previously mentioned in the first chapter, whose activities are tied to logistic analysis, architectural planning, or the delivery of their services, may be interested in the proposed solution. Effective implementation of the program can attract public investment and contracts, which will also affect the project's payback. The most interested in using this software will be countries with a small area that should be used in the most optimal way, including the Republic of Moldova, which may additionally be interested in using software developed in the domestic market (for example, for national security purposes).

According to the information provided, it is noticeable the multiple directions of the program application areas, which gives a wide audience coverage. With high-quality marketing, the program has the opportunity to become a profitable project. There are many clients on the market and, due to the latest market trends, their number is growing, which creates a favorable environment for entering the market. Also, the lack of alternatives greatly improves the likelihood of project success.

### **4.2.1. Customer avatar**

To improve product promotion on the market, it is worth making a rough portrait of a potential client, but the difficulty lies in targeting the software at legal entities and government agencies. Here, an attempt will be made to analyze a potential buyer, considering the possible influence of certain individuals on the purchase of this software.

Consideration of individual enterprises as buyers is unlikely due to the lack of several service points of such companies, however, it is possible to consider individual entrepreneurs engaged in cartographic analysis or logistics planning, for whom the use of this software can be an effective and profitable tool.

Cartographers are specialists in a complex field that requires broad knowledge of mathematics, geography, physics and other sciences, which brings the client area under consideration towards middle-aged people with higher education and due to the spread of the direction of work with very different salaries, including low-paid specialists [45].

Logisticians, in turn, are specialists in the more narrowly focused area of ​​optimizing logistics processes and structures within companies or organizations, which requires a specialist with higher economic education, organizational skills and experience to be able to work in basic logistics positions. Because of these factors, the potential client in this field is a middle-aged college-educated person with an average salary [46].

The use of this software within IT companies is possible in case of interest of system architects and leading developers of programs, applications and systems with a distributed computing system, a multitude of computational modules at their disposal and the need to optimize the load on these modules. To obtain the status of a leading developer or software architect, person must have experience in this industry, experience in software design, familiarity with the infrastructure of large IT companies. These factors determine these employees as key in making decisions on the acquisition of technologies for building their software and, on average, this person will be a middle-aged person with extensive work experience [47].

The use of the program in more traditional forms of doing business will be applied at the managerial level, which entails the need to attract the attention of managers, employees of executive positions. The use of this program companies that have an existing network of service points will be especially significant.

Within government agencies, the situation is more difficult because it will involve drawing attention from people in charge of government services and ministries, which means attracting attention from ministers and officials. Due to the complexity of career growth and the complex work process, getting this position most often occurs in middle-aged people.

It is noticeable that the avatar of a potential user is linked to the position he occupies within the company or to the activity being carried out. For this reason, in order to promote the program and familiarize potential users with them, it is worth highlighting social networks based on professional activities, postal services of enterprises for communication on business issues, promotion in search services in professional areas and the status of a person requesting information.

## **4.3. SWOT analysis**

SWOT analysis is an effective tool for analyzing internal and external factors of an organization or a product, dividing factors into several categories, namely, strengths, weaknesses, opportunities and threats. The first two categories are posed for analyzing internal factors and characteristics, and the remaining two are posed for examining external factors [48].

When properly constructed, this analysis tool will allow user to objectively consider the product or work of the organization for further improvement, compensating for weak points or preparing for negative factors and aspects. Below is a table of SWOT analysis.

Table 4.2 - swot analysis (made by author)

|  |  |
| --- | --- |
| Strengths | Weaknesses |
| - The simplicity of the program structure makes it possible to adapt it both for local use (application) and as a microservice or web application;  - The cheapness of the implementation of the program;  - Low requirements for computing resources;  - Ease of use of the program | - Direct implementation of the program into complex cartographic or navigation systems can be a difficult process that requires additional resources;  - Dependence of the program on the use of static map services;  - If there is a constant movement of service points with dynamic display of optimal zones in real time, introduction of additional algorithms (for example, calibration of incoming data) will be required;  - Low error in converting points from one coordinate system to another and plotting a diagram over the map area |
| Opportunities | Threats |
| - Large potential client base and the lack of alternatives;  - The structure of the program allows its further restructuring and improvement depending on the needs of the market;  - User base extension with new features and improvements;  - The low demand of the project for resources increases the attractiveness of the project for investors and minimizes potential losses. | - Implementation of this solution within a larger system or program (for example, the implementation of this functionality in Google Maps);  - Implementation of a full-fledged alternative with large investments;  - Low efficiency of promotion and advertising;  - Demanding a larger set of functions and capabilities from customers. |

Google maps and Yandex maps provide algorithms for finding the route of the minimum distance or minimum cost in time. These systems work, but they are clumsy: the optimality of the choice of a service point is not considered, its state, many points are absent on the map, and their implementation / change goes through a long moderation system [49].

When the functions and capabilities of the program under consideration are introduced into the above systems, the attractiveness of the program for potential customers will significantly decrease. It is important to note that such a change is unlikely due to the complexity of integrating such solutions into large systems, however, even in such a scenario, it is possible to compete due to the ability to track changes in real time, the ability to move points of interest. The creation of a complete alternative solution by a large company is a more significant risk.

The main risk is a decrease in the cost of the ArcGIS platform for cartographic research, or the provision of a narrower set of tools at a reduced cost. In such a situation, the attractiveness of the software in question will be at a low level and the lack of funding with promotion can lead to losses from the implementation of the project. It is important to note that this outcome is unlikely due to the long period of presence of the ArcGIS platform on the market and the lack of action to occupy new market areas due to the occupied niche. Despite the availability of opportunities for such a long time, ESRI has not developed highly targeted programs for an easier acquisition of the program, which indicates the company's lack of interest in this direction.

The problem is also the need to advertise the program and its correct positioning in the market, which can be difficult due to its narrow focus. The client coverage of the program directly depends on the promotion of the program in the market due to the need to inform about the availability of this solution. Despite solving a significant problem, the client base may remain uninterested in the program due to the lack of additional features.

Despite the risks and weaknesses of the project, the opportunities and strengths outweigh the negative sides and the project has a chance to become profitable. The most important aspect of the project is its simplicity and unusual possibilities, which allows it to occupy its own niche in the market and attract investors interested in a project that is low-risk in terms of losses, but attractive in terms of opportunities.

## **4.4. Time management and estimations**

Since further a detailed work plan will be considered with the definition of dependencies, sequences and with a visualization of the planned structure of work, the review will be carried out on the basis of the main stages of development to each milestone with commenting on important points.

### **4.4.1. Gantt chart**

To set up detailed planning with prioritization, specifying dependencies, sequences and deadlines, the Gantt chart will be used as an effective project planning tool [50]. Each big program stage will be presented with parts of Gantt chart that show dependencies and sequences of tasks.

The initial stage for starting work is the design of the application, the definition of the approximate architecture of the software being developed. During the work over this section, the team will have to identify the most important points in the development, and determine the main stages of future work. For understanding, it is worth decomposing this section into components and present graphical representation of tasks shown in the figure 4.1.

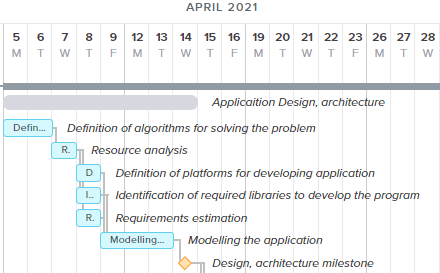


Figure 4.1 - application design and architecture estimation stage (made by author)

Determination of algorithms that allow solving the problem is a fundamental step in further planning and therefore the entire development team will be involved at this stage. After establishing the algorithms and principles for solving the problem, the team can establish resource requirements for the software, indicating the data to be operated and their volumes, setting the limits of the required computing power, and so on. After this step, it is possible to split the team into the following tasks for parallel execution: defining platforms for development, setting requirements for the developed program to set quality criteria and adherence to the specified metrics to obtain an effective solution upon completion of development, determining the technologies and libraries used to solve the task in the most efficient way. After completing work with these processes, the development team can start modeling the system for transparency of further development and adherence to the specified constraints and descriptions.

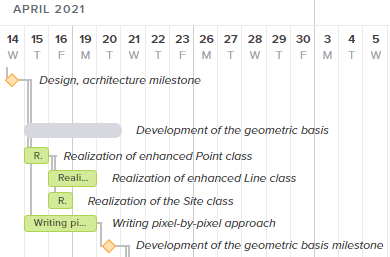


Figure 4.2 - development of the geometric basis (made by author)

The next stage of development is to develop a geometric basis for further building diagrams (coverage areas), visualization of which is presented in the figure 4.2. During this part of the project, the team will be broken down into the following tasks: one part of the team develops the foundations of the geometric class Point, after which it is divided into two more parts, which are simultaneously developing the class Site responsible for storing information on the service point, and the implementation of the geometric class Line, and the second part deals with parallel implementation of the pixel-by-pixel method for constructing coverage zones for the possibility of further checking the correctness of the defined zones when implementing other, more complex algorithms.

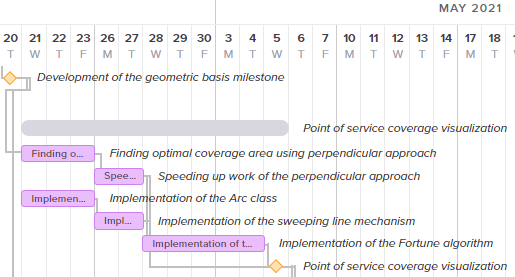


Figure 4.3 - service point coverage visualization mechanism development stage (made by author)

Visualizing service point coverage areas is a complex task that will consist of several stages (Gantt chart part for this stage is shown in the figure 4.3). To begin with, a solution based on the perpendicular method and a solution based on Fortune's method will be implemented on a parallel basis. The final stage of developing Fortune's methodology, which consists in building an algorithm, is carried out by the whole team together after the development of the perpendicular method is completed and the Ark class is written with the sweeping line mechanism. If the development is carried out with a delay, then it is possible to remove the entire algorithm for developing Fortune's methodology from the set of realizable capabilities with preparatory stages such as the implementation of the Ark class and the sweeping line mechanism.

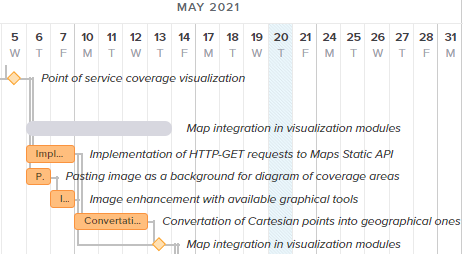


Figure 4.4 - Map integration and visualization modules development stage (made by author)

The next step is to add map support to plot the chart over the real geographic area. Graphical representation of this stage structure is shown in the figure 4.4. At the same time, tasks are performed to set the picture as the background of the diagram, write algorithms for improving the picture (such as applying anti-aliasing) and implementing the receipt of the necessary fragment of the map through HTTP-GET requests. After completing these stages, the team takes on the operations of converting the coordinates of points from one system to another, which cannot be done without understanding the principles by which a fragment of the map is cropped and inserted into the background.

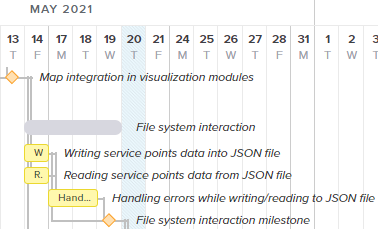


Figure 4.5 - File system interaction development stage (made by author)

The implementation phase of interacting with the file system begins with the parallel implementation of writes to a service point file and reads from a service point file (visualization of current stage structure is shown in the figure 4.5). After the completion of the implementation of these operations, the algorithms for working with the file system are being improved to increase the stability of the application. In further versions and releases program can be enhanced to support additional file types, improvement of overwriting and overreading principle, making it adaptable to choices of the user.

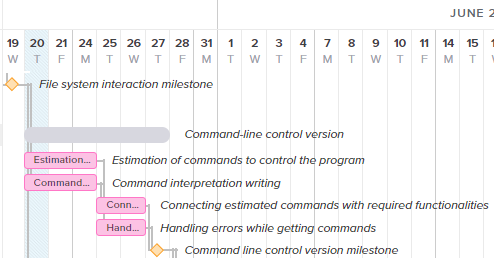


Figure 4.6 - Command line control version development stage (made by author)

After the implementation of this operation, the stage of developing a version of the program for use in the command line mode begins with structure of the stage shown in the figure 4.6. To do this, a command divided into two parts in parallel is busy first with defining commands and passed arguments with the creation of a command interpreter, and then busy with connecting commands, a command interpreter, binding functions to commands and increasing the stability of this version by catching errors.

The last stage of software development will be the implementation of a graphical user interface for working with the program, for which the team will be divided into many for parallel execution of tasks conform figure 4.7.

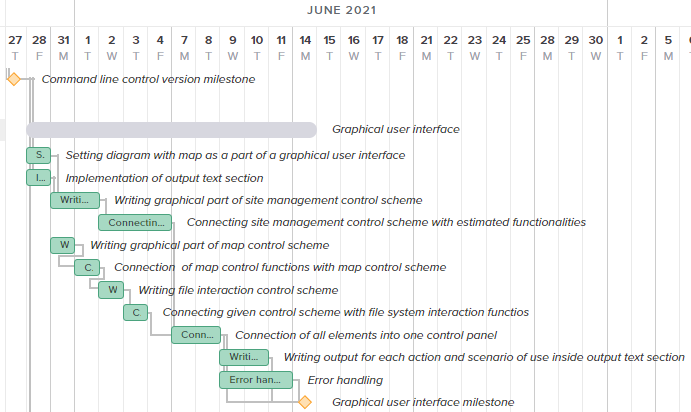


Figure 4.7 - Graphical User Interface development stage (made by author)

The first phase of execution will be the adaptation of the displayed diagram as a part (section) of the user interface with the implementation of the text output section. The second phase will be to split the team into two parts, where one will be busy with the implementation of the control panel for the points of service, and the second will take over the implementation of the panels for editing the map and interacting with the file system. The third phase will be the connection of all panels into a single graphical entity with an increase in the stability of the program by catching errors.

In case of unforeseen circumstances, it is possible to refuse to implement one of the developed versions. With a maximum lack of time, it is possible to refuse to implement a version with a graphical interface, however, with less urgency and more desire, the team can only implement the management of the graphical interface. If one of the control methods is not implemented, it can be implemented in future updates with the support of the program.

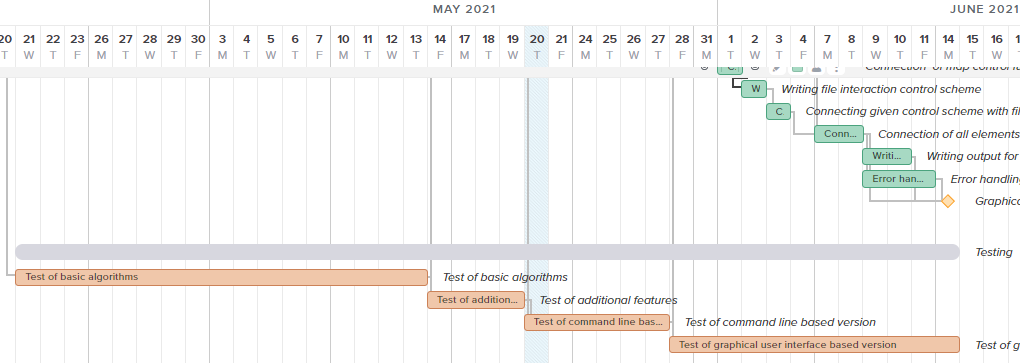


Figure 4.8 - Testing development stage (made by author)

Throughout the development cycle, starting from the stage of implementation of the fundamental capabilities (basis) of the program, testing of the program will be carried out with the subsequent elimination of detected errors and malfunctions conform figure 4.8. Such a structure will allow at the development stage to eliminate errors so that they cannot be transferred from one version of the program to another and to eliminate them at the early stages of their appearance before their possible impact on other elements of the program begins.

Due to the complexity of the development and the long implementation period, it is recommended to use a team of 4 developers to effectively develop the program. For testing are recommended 2 testers.

To effectively lead the development process, a development and test manager is needed.

When considering the general stages of project implementation, it is worth considering the general table of the planned time without decomposition into small tasks.

Table 4.3: general time estimations (made by author)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stage nr. | Stage description | Period | | Duration (working days) |
| Begin | End |
| 1 | Market research and topic definition | 19.03.2021 | 23.03.2021 | 3 |
| 2 | Definition of tasks and characteristics of the project | 24.03.2021 | 29.03.2021 | 4 |
| 3 | Domain analysis, analysis of project positions in the market | 30.03.2021 | 05.04.2021 | 5 |
| 4 | Program design | 05.04.2021 | 14.04.2021 | 8 |
| 5 | Program development and testing | 14.04.2021 | 14.06.2021 | 49 |
| 6 | Documenting the program | 15.06.2021 | 17.06.2021 | 3 |
| 7 | Program release and launch | 18.06.2021 | 21.06.2021 | 2 |

Based on the considered time frame, can be determined the costs of the project.

### **4.4.2. Scrum project management**

Due to the small size of the development team, the planned implementation structure divided into several sections and the need to adapt the development process for possibly changing sets of tasks, filter out less priority tasks and have a more or less complete product by the end of development, it was decided to use Scrum management as the main method of development [51], detailed schema of which is shown in the figure 4.9.

All major stages of development will be converted into sprints so that stages completed in several days will be combined into one major sprint, and longer stages are sprints in themselves. To simplify the structure of the considered project as startup, the director of the company can be considered as a product owner, however, when implementing a project within a large development company, this position will be occupied by a client-customer.

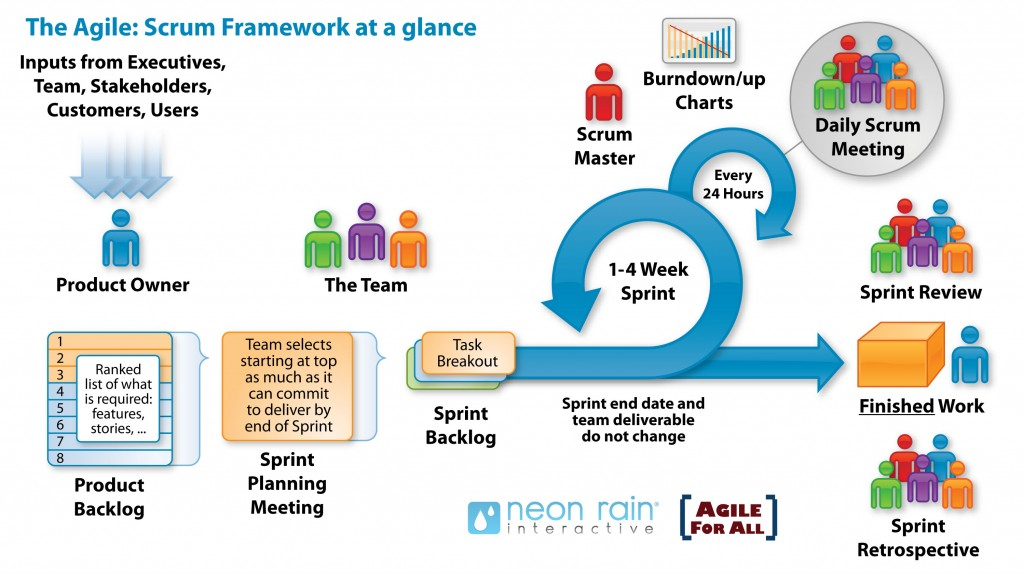


Figure 4.9 - scrum project management schematic explanation (taken from source [52])

For this reason, the project manager must be familiar with Scrum development methodologies and the ability to effectively organize the workflow. If the manager has access, it is recommended to use auxiliary tools for organizing Scrum project management.

## **4.5. Costs and profit estimation**

Assessing the attractiveness of the project for investors and the possibilities for its implementation, it will be considered as a startup. The costs will be considered sequentially and will subsequently lead to the overall picture of the project. To begin with, it is worth considering the table of calculations for wages and the number of required labor resources.

### **4.5.1. Labor costs**

The below table shows the employee cost to implement a working version of the program that can be released for release. This ignores the support costs associated with initializing the company, purchasing hardware, and post-release program support at the moment.

The developer undertakes to develop the basic algorithms for the program, its implementation from a computational point of view. The tester will check the functions of the program for errors, inaccuracies or irregularities in their work. The project manager will be in charge of managing the activities of these employees, organizing their work and setting tasks. If necessary, the manager can also take part in the development of the project.

The company's director and accountant will take over the administrative functions of the company. The director can also undertake obligations to conclude the agreements required for the company.

Table 4.4 - table of costs for development process, MDL (made by author)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Employee qualification | Employee amount | Required working hours | Salary per hour | Accrued salary | Health insurance | Social insurance | Income tax | All costs | Net salary |
|
| Developer | 4 | 392 | 125,00 | 49 000,00 | 4 410,00 | 11 760,00 | 5 098,80 | 60 760,00 | 39 491,20 |
| Tester | 2 | 312 | 90,00 | 28 080,00 | 2 527,20 | 6 739,20 | 2 814,34 | 34 819,20 | 22 738,46 |
| Manager | 1 | 392 | 125,00 | 49 000,00 | 4 410,00 | 11 760,00 | 5 098,80 | 60 760,00 | 39 491,20 |
| Accountant | 1 | 196 | 125,00 | 24 500,00 | 2 205,00 | 5 880,00 | 2 423,40 | 30 380,00 | 19 871,60 |
| Director | 1 | 196 | 150,00 | 29 400,00 | 2 646,00 | 7 056,00 | 2 958,48 | 36 456,00 | 23 795,52 |
| All workers | 9 |  |  | 179 980,00 | 16 198,20 | 43 195,20 | 18 393,82 | 223 175,20 | 145 387,98 |

To determine the amount of working time for the period of software development, the number of working days calculated within the planned time frame was taken with a daily rate of 8 hours. Based on these calculations, the total required number of working hours was determined. The calculation of costs also considers the payment of social and health insurance taxes.

Social Security Tax:

(4.1)

Health Insurance Tax:

(4.2)

Social insurance is paid by the employer, while health insurance is taken from the employee's salary. In addition, it is worth considering the payment of income tax from an individual in the amount of 12%, which is calculated using the following formula (income tax is levied on the salary of an individual):

(4.3)

Determining the costs of software support at the moment is not possible, since the client ordering this project may change his requirements for the final program. When self-implementing a project, forecasting costs is also not possible and only the implementation of the program to a working and acceptable build is taken for consideration, which will be finalized based on user interest.

### **4.5.2. The costs of opening and monthly maintenance of the company**

For software development, planning and documentation management, the company will use solutions free of charge, which will save on software development due to limited financial resources. For development, the integrated development environment Intellij IDEA Community Edition, the text editor Visual Studio Code, the version control utility Git, as well as office applications from Google or LibreOffice will be used.

Table 4.5 - costs of opening company and starting work, MDL (made by author)

|  |  |
| --- | --- |
| Company registration (considered use L.L.C.) | 1 150,00 |
| Personal laptops for workers in the company | 84 510,00 |
| Monitors (screens) for employees | 27 711,00 |
| Projector | 7 438,00 |
| Consumables | 2 500,00 |
| Overall costs | 123 309,00 |

It is important to emphasize that for the possibility of full-fledged development, employees will be provided with laptops for work, monitors for convenient work, a projector for presentations [53] [54] [55]. For efficiency and cost optimization, a small office with furniture will be used for the absence of furniture purchase costs.

Additional costs include office rent [56], internet [57], advertising costs, and minor ancillary costs.

Table 4.6 - table of costs for each month of work of company (made by author)

|  |  |
| --- | --- |
| Office rent | 6 850,00 |
| Internet | 100,00 |
| Consumables | 200,00 |
| Advertising costs | 21 000,00 |
| Overall costs | 29 950,00 |

It is important to pay attention to marking costs in order to understand their value and how it was formed basing on a small research.

### **4.5.3. Marketing costs**

Advertising on the search network can be used to promote the program, which can produce effective marketing results. The minimum rate is from 6 000 MDL [58]. The cost of costs in the search network is so significant, because with a previously defined avatar of the buyer, it was possible to identify individuals who may be interested in purchasing or significantly influence the purchase of the specified product by the company. For these purposes, it is necessary to use a targeted advertising system based on user requests (or query topics), as well as on the basis of user activity areas, which can be set when implementing a marketing campaign on search engine resources.

To promote the program on social networks for workers and oriented on professional solutions (like LinkedIn), the costs will amount to about 15 000 MDL, which will attract clients to the web-page in this social media [59]. The approximate cost of advertising within this network is determined by the minimum cost per click of two dollars. When converting the amount of 15,000 MDL into US dollars, can be seen that the approximate rate will be 830 dollars at the time of writing this thesis [60], which means attracting the attention of 415 potential clients within a month. It is worth noting again: this is an indicator of potential customers who, being interested in the viewed ad, will click on the corresponding page of the company.

Based on these rates, an amount for marketing of approximately 21 000 MDL is obtained.

### **4.5.4. Profit estimations**

Based on the considered costs and their amount, it is possible to compile the approximate cost of implementing this project as a startup. For the correct calculation, it is required to take the cost of starting a company (starting an activity), take the rate of monthly costs for the program and multiply it by the duration of the project (which in this case is 4 months) and add it to the total calculated costs of remuneration of all employees for a set period time. Thus, the total cost will amount to 446 284.20 MDL.

If the project is sold upon completion of its implementation, the bottom-up method will be used to determine the sale value. The desired profit is 30 000 MDL. The cost of the project is taken as the cost of its development, which is estimated at 446 284.20 MDL.

(4.4)

It is worth determining the ratio of profit to costs:

(4.5)

Based on this information, it is possible to calculate the cost of selling the project:

(4.6)

At this cost the project can be sold.

In the case of the sale of licenses for the use of this software, one by one, two monetization systems are allowed. Due to the possibility of distributing this software among legal entities and a narrow set of individuals, the price may be set at a low level (also for the possibility of competition with larger analogues). For this reason, when the price of a one-time purchase is set at 600 MDL, the program will be able to pay off and provide a profit when it is sold to at least 1000 persons, legal entities or individuals. When installing a subscription system, the cost can be 60 MDL per month, which will require purchasing a monthly subscription at least 10,000 times.

Such a pricing system will make it possible to recoup the project and compete with solutions presented on the market.

### **4.5.5. Further development with financial success and investment**

Upon receipt of financing, additional investments will be effectively directed to additional promotion of goods on the Internet. For these purposes, it is possible to develop website for the program with additional hosting costs [61], attract third-party advertising companies for a more aggressive promotion of advertising on the Internet.

To attract new categories of customers, an effective investment can be the development of additional features of the program, such as determining the optimal location of the service point depending on the infrastructure within the region under consideration and other possibilities.

In the future, upon receiving high recognition from clients and obtaining sufficient funding for the implementation of a large project, it is possible to implement its own cartographic system, which will allow developing its own set of cartographic tools and compete with large services like ArcGIS. It is also possible to conclude contracts, agreements on cooperation with large cartographic services for the joint implementation of a more complex system of cartographic analysis, which at the same time will lead to no need for financial resources due to the receipt of third-party funding.

When concluding agreements on cooperation with government agencies and services, a similar conclusion of agreements for the use of government resources or support programs for improving software is possible.

To save resources with further improvement of the project, it is also allowed to use third-party cartographic technologies on the basis of contracts and agreements. It is worth considering the potential loss of part of the income from the conclusion of this agreement due to the need to provide part of the profit to the company providing its technologies on the basis of the terms of the contracts.

## **Conclusion**

The economic analysis of the project demonstrated the possibility of its implementation with a profit, regardless of the format of its full sale to a third-party company or the format of the independent implementation of software as a product. Despite the narrow focus of the problem that the developed program eliminates, the audience coverage is wide and the program can become a profitable investment. It is worth noting the possibility of selling this product both on the local market and on the international market.

Due to the analysis of third-party solutions and competitors presented on the market, it is noticeable that the developed program will occupy its niche in the market and its further development to a large project that can influence the market of the provided cartographic services and goods.

Since the implementation of the project will be carried out according to forecasts by a small development team, it is recommended to use Scrum project management techniques for management. Task planning was carried out considering the possibility of abandoning the least demanded tasks while reducing the project implementation time, delaying the development team, as well as considering the prioritization of tasks and building their sequences.

Despite the optimization of project costs when it is implemented as a startup, additional cost reduction is permissible, however, with such a strategy, there is a risk of reducing the quality of the software being developed, increasing development time and reducing the efficiency of employees.

Due to the low cost of the project, there is the possibility of selling it at a low cost to attract additional attention from the client base, and the possibility of cooperation with certain categories of clients can provide beneficial competitive advantages. Also, due to its low cost, the project may be of interest to larger cartographic companies, which may be attracted either by the prospect of using their cartographic services, thereby providing a share of sales, or by implementing software within their cartographic service.

The realized product has high economic potential and needs to be financed. Due to the complexity of the topic under consideration, the market lacks competition to provide business-friendly solutions of this kind and the current project could improve this situation. Despite the potential difficulties, the project may be of interest to both representatives of the local market and representatives of the foreign market. This project can be challenging for a small company, but when implemented in a large company with sufficient funding or with sufficient investment, the project can expand to provide a wide range of mapping and economic services.

# **CONCLUSIONS**

Initially, this thesis was considered as a scientific research aimed at solving a narrowly focused topic. Gradually, in the course of the development, the importance of the problem of determining the coverage areas of service points for the economy became obvious, since an effective solution to this issue would significantly improve the work of economic agents. Despite the importance of this topic, there are only a few services on the market that can partially solve this problem and only one tool that allows user to solve it fully, but its cost and complexity in use will not allow effectively and quickly solve the tasks with its help.

For this reason, the project, originally conceived as an algorithmic study, has grown into a complex document that considers not only the issues of mathematical and algorithmic implementation of software, but also a comprehensive analysis of the economic and social impact of the implementation of this project as a commercial product.

As part of the development, it was possible to implement high-performance and easy-to-use software that can be used in different versions depending on the user's needs. The developed program can be used both separately and as part of another program or system. Work was done to implement an efficient and easy-to-understand principle of the program's operation at the source code level with the possibility of its further improvement and the introduction of new features, such as considering third-party factors affecting the coverage area of ​​service points. This program is not without drawbacks and can be improved, however, the development of this program was also limited by the time frame and implementation by one student. It should also be borne in mind that despite the shortcomings, this software can already be used to solve the problem of determining the coverage areas of service points indicated in the thesis.

As part of the writing of this thesis, work was done on the analysis of theoretical sources, the transformation of the information received into algorithmic information and subsequently into the source code of the program, on the modeling and design of the developed program. Upon completion of the writing of this work, a complex and multi-level study was obtained. Conform results of economic research shown in the last chapter, implemented product can be competitive in the market, with an attractive concept for obtaining investments and prospects to become a popular solution.

The developed program can become the basis for further research as part of the master's work and become the basis for obtaining a more comprehensive, high-quality and versatile solution. Completion of the program can lead to the appearance on the market of a high-quality product of light cartographic analysis to optimize economic and logistic processes within enterprises.

# **BIBLIOGRAPHY**

1. YUMASHEV‌, A. *The 5 Best Things about Customer Service Speed.* Jitbit.com, ed. 8 October 2018 [Accessed March 31, 2021]. Available**:** [https://www.jitbit.com/news/3290-the-5-best-things-about-customer-service-speed/#:~:text=Speed%20in%20customer%20support%20establishes](https://www.jitbit.com/news/3290-the-5-best-things-about-customer-service-speed/" \l ":~:text=Speed%20in%20customer%20support%20establishes)
2. *Coronavirus’ Business Impact: Evolving Perspective.* McKinsey, 2020 [Accessed March 31, 2021]. Available:[https://www.mckinsey.com/business-functions/risk/our-insights/covid-19-implications-for-business#](https://www.mckinsey.com/business-functions/risk/our-insights/covid-19-implications-for-business)
3. ‌*6 Ways the Pandemic Has Changed Businesses*. World Economic Forum, McKinsey, November 13, 2020 [Accessed May 21, 2021] Available: <https://www.weforum.org/agenda/2020/11/covid-19-innovation-business-healthcare-restaurants/>
4. LORA, J.‌, PALUMBO, D., BROWN, D. *Eight Charts on How Coronavirus Has Hit the Global Economy.* BBC News, June 29, 2020 [Accessed March 31, 2021]. Available: <https://www.bbc.com/news/business-51706225>
5. VIRGILINTO, D. *Why Customer Service Needs to Be Faster than Ever.* Shopify Enterprise Ecommerce Blog, February 29, 2016 [Accessed March 31, 2021]. Available: <https://www.shopify.com/enterprise/94678726-the-need-for-speed-why-customer-service-needs-to-be-faster-than-ever>
6. MACDONALD, S. *5 Ways to Reduce Customer Service Response Times.* CRM Blog, August 13, 2019, ed. 2 March 2021 [Accessed March 31, 2021]. Available: <https://www.superoffice.com/blog/response-times/>
7. KOSTIN, E. ‌*Скорость работы сайта: что на неё влияет и на что влияет она сама?.* Ашманов и партнеры, ed. 2 October 2019 [Accessed March 31, 2021]. Available: <https://www.ashmanov.com/education/articles/skorost-raboty-sayta-chto-na-neye-vliyaet-i-na-chto-vliyaet-ona-sama/>
8. *What Is GIS? | Geographic Information System Mapping Technology.* Esri, 2019. [Accessed May 21, 2021]. Available: <https://www.esri.com/en-us/what-is-gis/overview>
9. ‌*Does Google Maps Use GIS? - Quora.* Quora [Accessed March 31, 2021]. Available: <https://www.quora.com/Does-Google-Maps-use-GIS>
10. ‌*Working with BigQuery GIS Data.* Google Cloud [Accessed March 31, 2021]. Available: <https://cloud.google.com/bigquery/docs/gis-data?hl=he>
11. ‌*Доставка еды, Доставка из ресторанов, Заказать еду онлайн - STRAUS.* Straus [Accessed March 31, 2021]. Available: <https://www.straus.md/ru/?filter=all>
12. *ArcGIS Online | Cloud-Based GIS Mapping Platform.* Esri, 2018 [Accessed May 21, 2021]. Available: <https://www.esri.com/en-us/arcgis/products/arcgis-online/overview>
13. MAGUIRE, D. *Dr. David Maguire on the ArcGIS 9.0 Product Family Release*. GISCafe, May 31, 2004 [Accessed May 21, 2021]. Available: <https://www10.giscafe.com/nbc/articles/view_weekly.php?articleid=208790>
14. *ArcGIS pro Pricing | Buy Now.* Esri, 2021 [Accessed May 21, 2021]. Available: <https://www.esri.com/en-us/arcgis/products/arcgis-pro/buy>
15. ‌*Service Area Analysis—ArcMap | Documentation*. ArcGIS, 2020 [Accessed May 21, 2021]. Available: <https://desktop.arcgis.com/en/arcmap/latest/extensions/network-analyst/servicearea.htm>
16. ANGEL, J. *What’s New in ArcGIS pro 2.5.* ArcGIS Blog, February 6, 2020 [Accessed May 21, 2021]. Available: <https://www.esri.com/arcgis-blog/products/arcgis-pro/announcements/whats-new-in-arcgis-pro-2-5/>
17. ‌‌JAMES, R. C. *Mathematics Dictionary*. Springer Science & Business Media, 1992. 225 p. ISBN 0-412-99031-8;
18. BURROUGH, P. A., MCDONNELL, R. A., LLOYD, C. D. *Principles of Geographical Information Systems. Third Edition*. OUP Oxford, 2015. pp. 160-161. ISBN 978-0-19-874284-5. Available: [https://books.google.md/books?id=kvoJCAAAQBAJ&pg=PA160&redir\_esc=y#v=onepage&q&f=false](https://books.google.md/books?id=kvoJCAAAQBAJ&pg=PA160&redir_esc=y" \l "v=onepage&q&f=false)
19. LASZLO, M. J. *Computational Geometry and Computer Graphics in C++*. Translated by V. L’VOV, Moscow: Binom, 1997, pp. 245-248. ISBN 5-7989-0008-8;
20. ‌ALEKSANDROV, A. D. *Математика, ее содержание, методы и значение. Том 3.* Book on Demand, Moscow, October 7, 2019. pp. 4-12. ISBN 978-5458255639
21. ‌DE BERG, M., DE KREVELD, M. OVERMARS, M., SCHWARTZKOPF, O. *Computational Geometry, 2-nd revised.* Springer-Verlag, 2000. pp. 151-160. ISBN 3-540-65620-0
22. ‌FORTUNE, S. *A sweepline algorithm for Voronoi diagrams // ACM Digital Library Proceedings of the second annual symposium on Computational geometry.* Assn for Computing Machinery, New York, 1986. pp. 313-322. ISBN 0-89791-194-6
23. ‌HEUNIS, J. *Fortune's Algorithm: An Intuitive Explanation.* Jacquesheunis.com, March 29, 2018 [Accessed April 28, 2021]. Available: <https://jacquesheunis.com/post/fortunes-algorithm/>
24. CORLAT, S., GREMALSCHI, A. *Metodologia rezolvării problemelor de geometrie computaţională.* Ministerul Educaţiei al Republicii Moldova Universitatea de Stat din Tiraspol, Chisinau, 2013. pp. 37-42. CZU 519.17+004.421.2(075.8)
25. ‌*Functional Requirements vs Non-Functional Requirements: Key Differences.* Guru99, August 31, 2019 [Accessed March 31, 2021]. Available: [https://www.guru99.com/functional-vs-non-functio nal-requirements.html](https://www.guru99.com/functional-vs-non-functio%20nal-requirements.html)
26. ‌"OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1” Object Management Group. 2011, p. 1. Retrieved 31 March 2021;
27. ‌MAU, Derek. *Guide to UML Diagramming and Database Modeling.* Microsoft, 24 September 2019 [Accessed March 31, 2021]. Available: <https://www.microsoft.com/en/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling>
28. ‌KRAUSE, E. F. *Taxicab Geometry: An adventure in non-Euclidean Geometry.* Dover Publications, New York, 1986. p.4. ISBN 0-486-25202-7
29. ‌Smith, Karl. 2013. *Precalculus: A Functional Approach to Graphing and Problem Solving*. *Google Books*. Jones & Bartlett Publishers, p. 8. ISBN 0-7637-5177-4;
30. ‌BLACK, P. E. *Manhattan Distance in Dictionary Of Algorithms and Data Structures.* Xlinux.nist.gov, ed. 11 February 2019 [Accessed March 31, 2021]. Available: <https://xlinux.nist.gov/dads/HTML/manhattanDistance.html>
31. *Slope-Intercept Form Introduction*. Khan Academy, 2018 [Accessed May 21, 2021]. Available: [https://www.khanacademy.org/math/algebra/x2f8bb11595b61c86:forms-of-linear-equations/x2f8b b11595b61c86:intro-to-slope-intercept-form/a/introduction-to-slope-intercept-form](https://www.khanacademy.org/math/algebra/x2f8bb11595b61c86:forms-of-linear-equations/x2f8b%20b11595b61c86:intro-to-slope-intercept-form/a/introduction-to-slope-intercept-form)
32. ‌*PassMark - AMD Ryzen 5 4500U*. PassMark Software, 2021 [Accessed May 18, 2021]. Available: <https://www.cpubenchmark.net/cpu.php?cpu=AMD+Ryzen+5+4500U&id=3702>
33. *PassMark - AMD Ryzen 5 3600*. PassMark Software, 2021 [Accessed May 18, 2021]. Available: <https://www.cpubenchmark.net/cpu.php?cpu=AMD+Ryzen+5+3600&id=3481>
34. *PassMark - Intel Core I7-7700HQ.* PassMark Software, 2021 [Accessed May 18, 2021]. Available: <https://www.cpubenchmark.net/cpu.php?cpu=Intel+Core+i77700HQ+%40+2.80GHz&id=2906>
35. *Сколько в Молдове предприятий.* Joblist.md, ed. 10 July 2019 [Accessed March 31, 2021]. Available: <https://joblist.md/ru/news/zakonodateljstvo/skolko-v-moldove-predpriiatii>
36. AKHALATSI, A., JOLEVSKI, F., ROVO, N. *Растущее воздействие COVID-19 на частный сектор Молдовы.* ed. Blogs.worldbank.org. 13 July 2020 [Accessed March 31, 2021]. Available: <https://blogs.worldbank.org/ru/europeandcentralasia/rastuschee-vozdeystvie-covid-19-nachastnyy-sektor-moldovy>
37. JASAL, S. *How COVID-19 Has Changed Business Dynamics, Requiring a Change in Approach.* YourStory.com, July 18, 2020 [Accessed March 31, 2021]. Available: <https://yourstory.com/2020/07/covid-19-business-impact-transition-strategy>
38. ‌WYLIE, C. *Six Ways Covid-19 Is Changing Business (for the Better).* The Drum, June 10, 2020.

Available: <https://www.thedrum.com/opinion/2020/06/10/six-ways-covid-19-changing-business-the-better>

1. ‌*Couriers & Local Delivery Services in the US - Number of Businesses 2005–2026.* IBISWorld, November 23, 2020 [Accessed March 31, 2021]. Available: <https://www.ibisworld.com/industry-statistics/number-of-businesses/couriers-local-delivery-services-united-states/>
2. MAZAREANU‌, E. *Topic: Trucking Industry in the U.S.* Statista, September 11, 2020 [Accessed March 31, 2021]. Available: <https://www.statista.com/topics/4912/trucking-industry-in-the-us/>
3. ‌*68 Useful ECommerce Statistics You Must Know in 2021.* WPForms, September 25, 2019 [Accessed March 31, 2021]. Available: [https://wpforms.com/ecommerce-statistics/#:~:text=There%20are%20 an%20estimated%2012](https://wpforms.com/ecommerce-statistics/" \l ":~:text=There%20are%20 an%20estimated%2012)
4. MLITZ, K. *Number of Data Centers Worldwide 2015-2021.* Statista, January 11, 2021 [Accessed March 31, 2021]. Available: <https://www.statista.com/statistics/500458/worldwide-datacenter-and-it-sites/>
5. ‌*Analysis Summary - BGP Routing Table Analysis - DIX-IE Data.* Apnic.net, March 31, 2021 [Accessed March 31, 2021]. Available: <http://thyme.rand.apnic.net/current/data-summary>
6. FERGAL, G. *Top 20 Logistics Companies.* 6 River Systems, February 9, 2021 [Accessed May 21, 2021]. Available: <https://6river.com/top-logistics-companies/>
7. YURKIN, A. *Зачем нужна профессия картографа?*. Kedu, 2021 [Accessed May 21, 2021].

Available: <https://kedu.ru/press-center/profgid/kartograf/>

1. *Профессия логист.* Enjoy-Job, 2021 [Accessed May 21, 2021]. Available: <https://enjoy-job.ru/professions/logist/>
2. PALKIN, A. *Кто такой системный архитектор и как им стать.* Tproger, September 11, 2020 [Accessed May 21, 2021]. Available: <https://tproger.ru/articles/becoming-a-system-architect/>
3. MAISAK, O. S. SWOT-анализ: объект, факторы, стратегии. Проблема поиска связей между факторами. CASPIAN JOURNAL: Management and High Technologies, *2013, no. 1 (21)*, pp. 151-157. UDC 339.138:65.012.123:004.4
4. ‌*Как подтвердить данные о местной компании в Google*. Google, 2021 [Accessed March 31, 2021]. Available: <https://support.google.com/business/answer/7107242?hl=ru&ref_topic=4854193>
5. RICHMAN, L. *Project Management Step-by-step.* Amacom, New York, 2002. 117 p. ISBN 978-0-8144-2657-9
6. SCWABER, K., SUTHERLAND, J. *The Scrum Guide the Definitive Guide to Scrum: The Rules of the Game.* Creative Commons, November 2020 [Accessed May 18, 2021].

Available:

<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>

1. PUSHKAREV, A. *Гибкая методология разработки ‘Scrum’.* Habr, January 4, 2015 [Accessed May 21, 2021]. Available: <https://habr.com/ru/post/247319/>
2. *Lenovo IdeaPad 3 15IIL05-ESRE I3-1005G1*. Enter, 2021 [Accessed May 21, 2021]. Available: <https://enter.online/ru/noutbuki/lenovo-ideapad-3-15iil05-esre-i3-1005g1-8-gb-256-gb-ssd-vga-integrated-seryy/>
3. *Проектор Asio LED RD826.* PandaShop, 2021 [Accessed May 21, 2021]. Available: <https://www.pandashop.md/ru/product/led-rd826-00235238/>
4. *Монитор LG 23.8* *24MK600M-W.* Enter, 2021 [Accessed May 21, 2021]. Available: <https://enter.online/ru/kompyutery/monitory/lg-23-8-full-hd-belyy-24mk600m-w/>
5. *Аренда офиса в Кишиневе на улице Матеевич, 1862.* Rental, 2021 [Accessed May 21, 2021].

Available: <https://rental.md/ru/birou-de-inchiriat-chisinau-1862/>

1. *Сервисы - StarNet.* Starnet, 2021 [Accessed May 21, 2021]. Available: <https://www.starnet.md/ru/services>
2. ‌*Заказать SEO продвижение сайта в Молдове, раскрутка сайтов.* Webmaster Studio, 2021 [Accessed March 31, 2021]. Available: <https://webmaster.md/uslugi/prodvijenie-saitov/>
3. GORDIENKO, M. *Таргетированная реклама в LinkedIN.* FreshIT, 2021 [Accessed May 21, 2021]. Available: <https://freshit.net/targetirovannaya-reklama-v-linkedin/>
4. Oanda *Currency Converter* [web-service], 2020 [Accessed May 20, 2021]. Available: <https://www1.oanda.com/currency/converter/>
5. *Веб-хостинг, услуги хостинга, VPS Хостинг в Молдове.* Tophost, 2021 [Accessed March 31, 2021]. Available: <https://tophost.md/ru/>

**CODE REPOSITORIES**

CRETU, D. *filpatterson/Voronoi\_Teza\_Project*. GitHub. April 27, 2021.

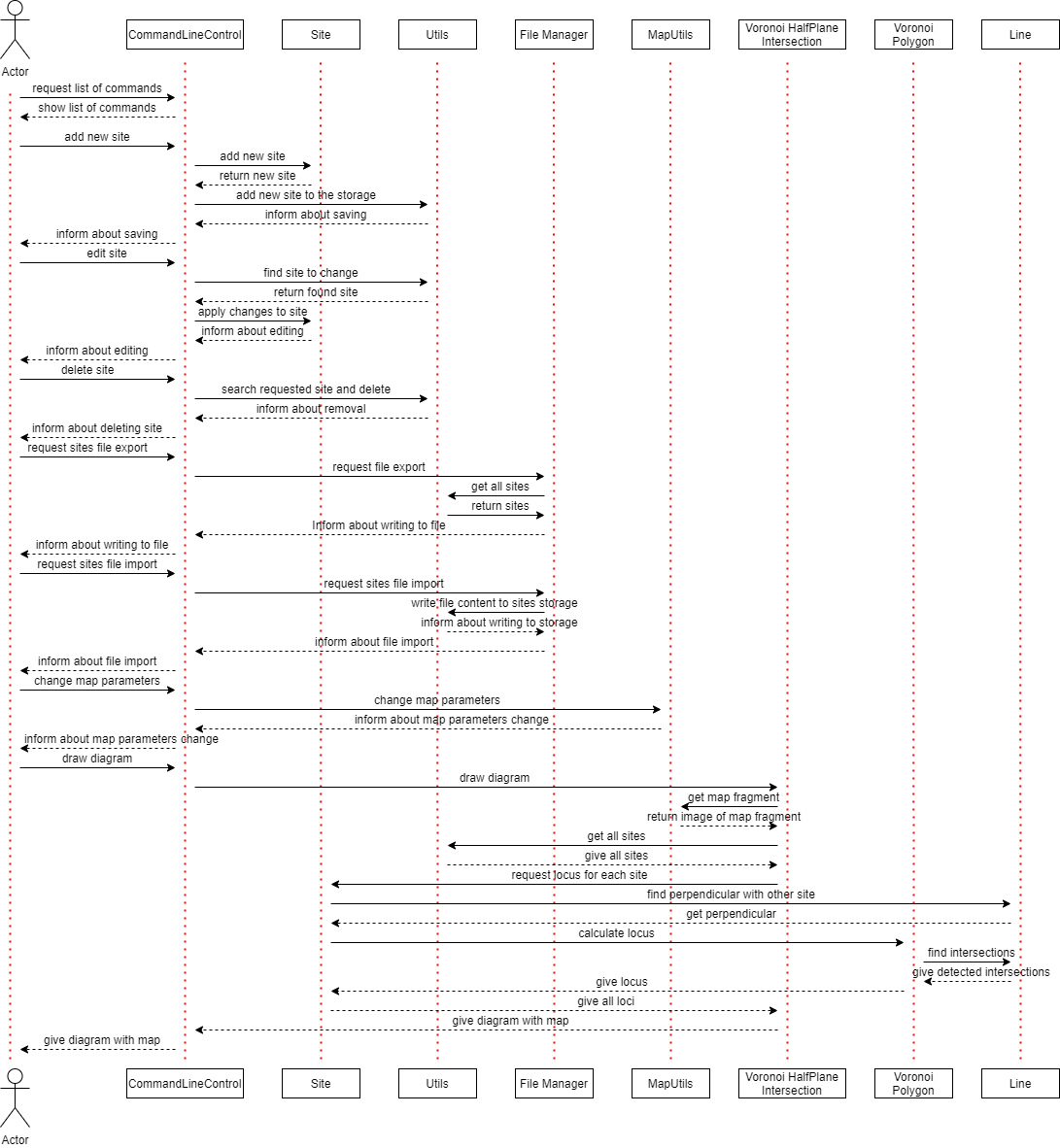
**Available:** <https://github.com/filpatterson/Voronoi_Teza_Project>

CRETU, D. *filpatterson/Voronoi\_teza\_legacy*. GitHub. May 2, 2021.

**Available:** <https://github.com/filpatterson/voronoi_teza_legacy>

# **ANNEXES**

**Annex nr. 1**



**Annex nr. 2**

|  |
| --- |
| /\*\*  \* constructor  \* **@param name** name of the figure  \* **@param interestPointsValue** how many interest points there are  \* **@param imageSize** size of section for PixelByPixelMethod.Voronoi Diagram  \*/  **public** VoronoiBrute(String name, **int** interestPointsValue, **int** imageSize) {  // set name of the graph  **super**(name);  // amount of interest points  **this**.interestPointsValue = interestPointsValue;  // set size of image  **this**.imageSize = imageSize;  // set window size and position relatively to the screen  setBounds(0, 0, **this**.imageSize, **this**.imageSize);  // set how program should be finished  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  // constructor of the image  **this**.image = **new** BufferedImage(**this**.imageSize, **this**.imageSize, BufferedImage.TYPE\_INT\_RGB);  // set arrays for storing X and Y coordinates of interest points  **this**.interestPointsX = **new int**[interestPointsValue];  **this**.interestPointsY = **new int**[interestPointsValue];  // set array of colors for interest points  **this**.interestPointsColors = **new int**[interestPointsValue];  } |

**Annex nr. 3**

|  |
| --- |
| /\*\*  \* generate locusts for all interest points (find nearest points) using colorization via pixel-by-pixel calculation  \* **@param isManhattanRequired** true if manhattan distance calculation is required, false if euclidean  \*/  **public void** voronoiLocustsIdentification(**boolean** isManhattanRequired) {  **long** startTime = System.currentTimeMillis();  // iterate through all "pixels" of the image  **for** (**int** currentPointX = 0; currentPointX < **this**.imageSize; currentPointX++)  **for** (**int** currentPointY = 0; currentPointY < **this**.imageSize; currentPointY++) {  **int** closestInterestPointIndex = 0;  // iterate through all cells of the image  **for** (**byte** currentInteresPoint = 0; currentInteresPoint < **this**.interestPointsValue; currentInteresPoint++)  // choose distance calculation method between Manhattan algorithm and Euclidean  **if**(isManhattanRequired) {  **if** (  manhattanDistance2D(  **this**.interestPointsX[currentInteresPoint], currentPointX,  **this**.interestPointsY[currentInteresPoint], currentPointY  ) < manhattanDistance2D(  **this**.interestPointsX[closestInterestPointIndex], currentPointX,  **this**.interestPointsY[closestInterestPointIndex], currentPointY)  ) {  closestInterestPointIndex = currentInteresPoint;  }  } **else if** (  euclideanDistance2D(  **this**.interestPointsX[currentInteresPoint], currentPointX,  **this**.interestPointsY[currentInteresPoint], currentPointY  ) < euclideanDistance2D(  **this**.interestPointsX[closestInterestPointIndex], currentPointX,  **this**.interestPointsY[closestInterestPointIndex], currentPointY)  ) {  closestInterestPointIndex = currentInteresPoint;  }  // apply color of the cell to current "pixel"  **this**.image.setRGB(currentPointX, currentPointY, interestPointsColors[closestInterestPointIndex]);  }  **long** endTime = System.currentTimeMillis();  System.out.println("Execution time is " + (endTime - startTime) + " ms.");  } |

**Annex nr. 4**

|  |
| --- |
| /\*\*  \* Finds perpendicular of the line basing on first degree polynomial calculations.  \* **@return** perpendicular of the line limited by area  \*/  **public** LineOld getPerpendicularByEquation() {  // if line is horizontal, then get vertical perpendicular  **if** (firstPointOld.getY() == secondPointOld.getY()) {  PointOld middle = middleOfLine();  **return new** LineOld(  **new** PointOld((**float**) middle.getX(), 0),  **new** PointOld((**float**) middle.getX(), ParametersOld.yLimit)  );  }  // if line is vertical, then get horizontal perpendicular  **else if** (firstPointOld.getX() == secondPointOld.getX()) {  PointOld middle = middleOfLine();  **return new** LineOld(  **new** PointOld(0, (**float**) middle.getY()),  **new** PointOld(ParametersOld.xLimit, (**float**) middle.getY())  );  }  // storage for perpendicular endpoints  ArrayList<PointOld> perpendicularEndpoints = **new** ArrayList<>();  **if** (**this**.m == 0 && **this**.b == 0) {  // find coefficient of the line, defining its "angle"  **this**.m = (**float**) ((secondPointOld.getY() - firstPointOld.getY()) /  (secondPointOld.getX() - firstPointOld.getX()));  **this**.b = (**float**) (secondPointOld.getY() - **this**.m \* secondPointOld.getX());  }  // find equation of the perpendicular  **double** perpM = -1 / **this**.m;  PointOld middle = middleOfLine();  **double** perpB = middle.getY() - perpM \* middle.getX();  // check if there is perpendicular endpoint on the left border  **double** yValueAtXBorder = perpM \* 0 + perpB;  **if** (yValueAtXBorder <= ParametersOld.yLimit && yValueAtXBorder >= 0) {  perpendicularEndpoints.add(**new** PointOld(0, (**int**) yValueAtXBorder));  }  // check if it is on the right border  yValueAtXBorder = perpM \* ParametersOld.xLimit + perpB;  **if** (yValueAtXBorder <= ParametersOld.yLimit && yValueAtXBorder >= 0) {  perpendicularEndpoints.add(**new** PointOld(ParametersOld.xLimit, (**int**)yValueAtXBorder));  }  // check if it is on the top border  **double** xValueAtYBorder = 0;  **if** (perpendicularEndpoints.size() < 2) {  xValueAtYBorder = (0 - perpB) / perpM;  **if** (xValueAtYBorder <= ParametersOld.xLimit && xValueAtYBorder >= 0)  perpendicularEndpoints.add(**new** PointOld((**int**) xValueAtYBorder, 0));  }  // check if it is on the bottom border  **if** (perpendicularEndpoints.size() < 2) {  xValueAtYBorder = (ParametersOld.yLimit - perpB) / perpM;  **if** (xValueAtYBorder <= ParametersOld.xLimit && xValueAtYBorder >= 0) {  perpendicularEndpoints.add(**new** PointOld((**int**) xValueAtYBorder, ParametersOld.yLimit));  }  }  **if** (perpendicularEndpoints.size() == 0) {  System.out.println("somethings wrong!");  }  // return perpendicular line  **return new** LineOld(perpendicularEndpoints.get(0), perpendicularEndpoints.get(1));  } |

**Annex nr. 5**

|  |
| --- |
| /\*\*  \* finds all corners of the halfplane by iterating through area borders (requires setting borders in clockwise direction)  \* **@param borders** borders of the area defined in clockwise direction  \* **@param perpendicular** perpendicular of the line  \* **@return** list of points defining halfplane corners  \*/  **private** ArrayList<PointOld> findCornersOfHalfplane(ArrayList<LineOld> borders, LineOld perpendicular, **boolean** isSecondOneRequired) {  // flag that will detect if half plane was completely found  **int** perpendicularPointsMet = 0;  // half plane corners array  ArrayList<PointOld> halfplaneCorners = **new** ArrayList<>();  **if** (!isSecondOneRequired) {  // iterate through sector borders (works with either clockwise or anti-clockwise direction)  **for** (LineOld border : borders) {  // if any of the perpendicular is met -> append it to half plane corners -> show this to the flag  **if** (border.containsByEquation(perpendicular.getFirstPointOld())) {  perpendicularPointsMet++;  halfplaneCorners.add(perpendicular.getFirstPointOld());  } **else if** (border.containsByEquation(perpendicular.getSecondPointOld())) {  perpendicularPointsMet++;  halfplaneCorners.add(perpendicular.getSecondPointOld());  }  // if both ends of perpendicular was checked -> half plane is found  **if** (perpendicularPointsMet == 2) {  **break**;  }  // if corner is a part of half plane -> append it to the half plane corners list  **else if** (perpendicularPointsMet == 1) {  halfplaneCorners.add(border.getSecondPointOld());  }  }  } **else** {  **for** (LineOld border : borders) {  // if corner is a part of half plane -> append it to the half plane corners list  **if** (perpendicularPointsMet == 0 || perpendicularPointsMet == 2) {  halfplaneCorners.add(border.getFirstPointOld());  }  // if any of the perpendicular is met -> append it to half plane corners -> show this to the flag  **if** (border.containsByEquation(perpendicular.getFirstPointOld())) {  perpendicularPointsMet++;  halfplaneCorners.add(perpendicular.getFirstPointOld());  } **else if** (border.containsByEquation(perpendicular.getSecondPointOld())) {  perpendicularPointsMet++;  halfplaneCorners.add(perpendicular.getSecondPointOld());  }  }  }  **return** halfplaneCorners;  } |

**Annex nr. 6**

|  |
| --- |
| /\*\*  \* Find half plane of this site using perpendicular estimated with another site  \* **@param perpendicular** perpendicular that was calculated between this site and another one  \* **@param useCustomSecondHalfEstimation** true if use of custom algorithm is required, false if required area negation  \* **@return** Half plane area containing this site  \*/  **private** Area findHalfPlane(LineOld perpendicular, **boolean** useCustomSecondHalfEstimation) {  // iterate through sector borders in clockwise direction  ArrayList<PointOld> halfplaneCorners = findCornersOfHalfplane(borders, perpendicular, **false**);  // form polygon out of estimated corners  Polygon halfPlane = **new** Polygon();  **for** (PointOld corner : halfplaneCorners) {  halfPlane.addPoint((**int**) corner.getX(), (**int**) corner.getY());  }  // if half plane contains site then return this area  **if** (halfPlane.contains(**this**.getX(), **this**.getY())) {  **return new** Area(halfPlane);  }  // if half plane does not have site then return another half plane from this sector  **else** {  **if** (useCustomSecondHalfEstimation) {  // use algorithm of finding corners of the halfplane with specified flag,  // defining which half plane is required and reset original halfplane  halfplaneCorners = findCornersOfHalfplane(borders, perpendicular, **true**);  halfPlane.reset();  **for** (PointOld corner : halfplaneCorners) {  halfPlane.addPoint((**int**) corner.getX(), (**int**) corner.getY());  }  **return new** Area(halfPlane);  } **else** {  // take area of sector and subtract from it area of the half plane that does not have site  Area siteArea = **new** Area(  **new** Rectangle(0, 0, ParametersOld.xLimit, ParametersOld.yLimit)  );  siteArea.subtract(**new** Area(halfPlane));  **return** siteArea;  }  }  } |

**Annex nr. 7**

|  |
| --- |
| /\*\*  \* find half polygon containing site by slicing polygon  \* **@param perpendicular** line that slices polygon  \* **@param site** point that must be inside polygon  \* **@return** half polygon with site inside  \*/  **public** VoronoiPolygon findHalfPolygon(Line perpendicular, Site site) {  // counter of found slices  **short** foundSlices = 0;  Point intersection;  // array list for vertices of half plane and new polygon reference  ArrayList<Point> newVertices = **new** ArrayList<>();  VoronoiPolygon foundHalfVoronoiPolygon;  **for** (Line edge : edges) {  // if current edge intersects line  **if** (edge.intersectsLine(perpendicular)) {  // find intersection point, add it to vertices list if no error detected and increment slice counter  intersection = edge.findIntersection(perpendicular);  **if** (intersection != **null**) {  newVertices.add(intersection);  foundSlices++;  }  **if** (foundSlices == 2) {  // create half polygon based on found vertices and check if site is inside  foundHalfVoronoiPolygon = **new** VoronoiPolygon(newVertices);  foundHalfVoronoiPolygon.setDrawable();  **if**(foundHalfVoronoiPolygon.contains(site)) {  **return** foundHalfVoronoiPolygon;  // else reset half polygon and start forming half polygon from second half  } **else** {  foundSlices++;  newVertices.clear();  newVertices.add(intersection);  }  }  }  // add each vertice that is a part of half polygon  **if** (foundSlices > 0) {  newVertices.add(edge.getSecondPoint());  }  }  // come again through starting point to form second half polygon  **for**(Line edge : edges) {  **if** (edge.intersectsLine(perpendicular)) {  intersection = edge.findIntersection(perpendicular);  **if** (intersection != **null**) {  newVertices.add(intersection);  **return new** VoronoiPolygon(newVertices);  }  }  newVertices.add(edge.getSecondPoint());  }  **return this**;  } |