Załącznik nr 7 do SIWZ

SZCZEGÓŁOWY OPIS PRZEDMIOTU ZAMÓWIENIA

1. PRZEDMIOT ZAMÓWIENIA

Przedmiotem zamówienia jest wykonanie usług informatycznych w zakresie projektu „Centrum Naukowych Analiz Geoprzestrzennych, Obliczeń Satelitarnych wraz z laboratoriami testowania/certyfikacji produktów geomatycznych (CENAGIS)", polegających na opracowaniu, wykonaniu i dostawie biblioteki wykonanej w technologii CUDA wraz z interfejsem w języku Python, zawierającej wyspecyfikowane algorytmy służące do obliczeń na chmurach punktów i danych rastrowych z wykorzystaniem procesora graficznego (dalej zwanego GPU, od ang. Graphics Processing Unit) oraz dostarczenia oprogramowania typu desktop do wizualizacji i analizy chmur punktów z wykorzystaniem GPU. Biblioteka powinna posiadać interfejs zapewniający obustronną komunikację między CUDA a Python, z wykorzystaniem technologii Apache Arrow, umożliwiający wykorzystanie algorytmów w języku CUDA w sposób wydajny dla danych przechowywanych w strukturach biblioteki Pandas i wykorzystanie ich w ramach platformy obliczeń rozproszonych Apache Spark.

Zadanie, będące przedmiotem zamówienia, jest współfinasowane ze środków Europejskiego Funduszu Rozwoju Regionalnego w ramach Osi Priorytetowej I „Wykorzystanie działalności badawczo-rozwojowej w gospodarce" Działania 1.1. „Działalność badawczo-rozwojowa jednostek naukowych" Regionalnego Programu Operacyjnego Województwa Mazowieckiego na lata 2014-2020.

2. ZAKRES PRZEDMIOTU ZAMÓWIENIA

A. BIBLIOTEKA DO OBLICZEŃ NA CHMURACH PUNKTÓW I DANYCH RASTROWYCH

WYMAGANIA OGÓLNE

1) Biblioteka ma umożliwiać wykonywanie obliczeń na GPU w dwóch trybach: podwójnej i pojedynczej precyzji. Nie przewiduje się szczegółowego specyfikowania przez użytkownika rodzaju danych na wejściu/wyjściu. Funkcja uruchomiona w trybie podwójnej precyzji ma przyjmować dane w postaci podwójnej precyzji i zwracać dane w postaci podwójnej precyzji i analogicznie w przypadku trybu pojedynczej precyzji.
2) Biblioteka ma pracować na danych wejściowych oraz zapisywać wyniki w postaci Pandas DataFrame, umożliwiając w ten sposób wykorzystanie biblioteki z poziomu funkcji zdefiniowanych przez użytkownika Pandas UDF (ang. Pandas User Defined Function) w ramach platformy Apache Spark.
3) Biblioteka powinna być zgodna z następującymi, wykorzystywanymi w CENAGIS wersjami bibliotek: Apache Arrow >=0.15.1 <=0.16.0, Apache Spark >=2.4.6 <3.0.0, Pandas >=1.1.0.
4) Do transferu danych między Python a CUDA biblioteka powinna wykorzystywać technologie Apache Arrow i wykorzystywać proste typy danych w celu zapewnienia maksymalnej, niewymagającej konwersji danych wydajności obustronnego przekazywania danych między Python i CUDA.
5) Funkcje zaimplementowane w CUDA powinny być zoptymalizowane do działania na kartach wykorzystywanych w CENAGIS tj. Nvidia Tesla T4 16 GB, posiadających Compute capability v7.5

6) Wszystkie algorytmy powinny obsługiwać wartość noData (m.in. umożliwiać jej zdefiniowanie).
7) Szczegółowy zakres parametrów (w tym maksymalne rozmiary zbiorów danych) dla poszczególnych algorytmów zostanie doprecyzowany na etapie realizacji zamówienia. Przy czym obliczenia mają być wykonywane na GPU przynajmniej dla algorytmów nr: 1, 2, 3, 4, 5, 10, 11, 12, 13, 14, 15, 16, 20. Jeśli na etapie realizacji zostanie ustalone, że dany algorytm będzie zaimplementowany na GPU to nie będzie dla niego wymagana osobna implementacji na CPU. Algorytmy nr: 1, 2, 3, 4, 15, 16, 20 poza wyspecyfikowaną wersją wieloparametrową (np. pozwalającą na zdefiniowanie sąsiedztwa) mają zostać również dostarczone jako metody wywołujące konkretne opcje (np. konkretny typ sąsiedztwa). Dla algorytmów nr 1, 2, 3, 4, 13, 14, 15, 16 jako dane wejściowe mogą być stosowane dwie chmury punktów, pierwszą definiującą punkty, na których mają być wykonane obliczenia i drugą definiującą punkty, na podstawie których mają zostać wyznaczone sąsiedztwa i wartości.

LICENCJA I PRAWA AUTORSKIE

Biblioteka wytworzona lub zakupiona na potrzeby realizacji zamówienia jest Oprogramowaniem Dedykowanym, do którego wykonawca posiada autorskie prawa majątkowe na wszystkich polach eksploatacji wymienionych w art. 50 i art. 74 ust. 4 ustawy o prawach autorskich i prawach pokrewnych, łącznie z prawem do udzielania zezwoleń na wykonywanie zależnego prawa autorskiego lub podlegające licencjom FLOSS (free libre/open source software).

Wykonawca oświadcza i gwarantuje, że jakiekolwiek utwory w rozumieniu art. 1 ustawy z 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych, zwanej „ustawą", przekazane Zamawiającemu w trakcie realizacji umowy, ani korzystanie z nich przez Zamawiającego lub inne podmioty zgodnie z umową, nie będą naruszać praw własności intelektualnej osób trzecich, w tym praw autorskich, patentów ani praw do baz danych. W przypadku wystąpienia przeciwko Zamawiającemu przez osobę trzecią z roszczeniami wynikającymi z naruszenia jej praw, z winy Wykonawcy, Wykonawca zobowiązany jest do ich zaspokojenia i zwolnienia Zamawiającego od obowiązku świadczeń z tego tytułu.

Wykonawca udzieli Zamawiającemu niewyłącznej licencji na korzystanie z oprogramowania oraz związanej z nim dokumentacji, w zakresie i na zasadach opisanych w Umowie.

W szczególności licencja będzie uprawniała Zamawiającego do korzystania z Oprogramowania w nieograniczonej liczbie instalacji, przez nieograniczoną liczbę użytkowników jednocześnie, bez ograniczeń terytorialnych i na czas nieoznaczony. W ramach udzielonej licencji Zamawiający upoważniony będzie do tworzenia opracowań i modyfikacji oprogramowania w nieograniczonym zakresie, w związku z czym Wykonawca przekaże Zamawiającemu kompletny kod źródłowy Oprogramowania. W zakresie autorskich praw zależnych Wykonawca zezwoli Zamawiającemu na korzystanie z i rozporządzanie opracowaniami Oprogramowania oraz udzieli prawa do zezwalania na korzystanie z i rozporządzanie opracowaniami Oprogramowania.

W szczególności Wykonawca udzieli Zamawiającemu licencji na wykorzystanie oprogramowania i towarzyszącej mu dokumentacji na polach eksploatacji wymienionych poniżej:

a) utrwalania oprogramowania w pamięci komputerów, w tym spełniających funkcje serwerów,
b) zwielokrotniania oprogramowania bez żadnych ograniczeń ilościowych, w pamięci komputera, jak i w sieciach multimedialnych, w tym typu internet i intranet, w szczególności

on-line, a także poprzez wydruk komputerowy, na każdym znanym w dacie podpisania niniejszej umowy nośniku,

c) rozpowszechniania utworu bez żadnych ograniczeń ilościowych, odrębnie lub w ramach utworów zbiorowych, w szczególności poprzez wprowadzanie do obrotu oryginału lub egzemplarzy, na których utwór lub jego fragmenty utrwalono (w szczególności utrwalonych technikami, o których mowa w lit. a),
d) przesyłania za pośrednictwem sieci multimedialnych, w szczególności internetu i intranetu, on-line, w ramach komunikacji na życzenie, w tym również publicznego udostępniania w taki sposób, aby każdy mógł mieć do oprogramowania czy jego fragmentu dostęp w miejscu i w czasie przez siebie wybranym,
e) tłumaczenia, przystosowywania, zmiany układu lub jakichkolwiek innych zmian w programie komputerowym, z zachowaniem praw osoby, która tych zmian dokonała,
f) rozpowszechniania, w tym użyczenia lub najmu, programu komputerowego lub jego kopii.

SPECYFIKACJA ALGORYTMÓW

Poniżej przedstawiono wymaganą, minimalną funkcjonalność jaka musi być realizowana przez dostarczoną bibliotekę. Podane nazwy, jak i podział na grupy są propozycją i mogą ulec zmianie na etapie realizacji zamówienia:

1) GETPROXIMITY

Algorytm, który dla każdego punktu zwraca zdefiniowaną strukturę danych zawierającą punkty znajdujące się w sąsiedztwie, jako sąsiedztwo przyjmowany może być:

- walec (definicja: promień),
- sfera (definicja: promień),
- graniastosłup o podstawie kwadratu (definicja: długość boku),
- wycinek pionowy walca (definicja: promień, wysokość),
- wycinek graniastosłupa o podstawie kwadratu (definicja: długość boku, wysokość),
- wycinek poziomy walca (kąt początku i kąt końca, promień, wysokość).

Rozmiar sąsiedztwa może być stały lub pochodzić z atrybutu, algorytm powinien pozwalać na określenie górnego limitu liczby punktów w sąsiedztwie jak i zdefiniowania sąsiedztwa jako:

- n najbliższych punktów w 2D,
- n najbliższych punktów w 3D.

2) GETSTAT

Algorytm, który oblicza wybraną statystykę (lub kilka statystyk naraz), przyjmując jako parametry definicję sąsiedztwa (z GetProximity) lub strukturę danych zawierającą predefiniowane punkty w sąsiedztwie, predefiniowane statystki to:

- wartość minimalna,
- wartość maksymalna (dla min i max dodatkowo możliwość policzenia kolejnych n wartości min/max),
- średnia,
- średnia kwadratowa,
- średnia ważona (IDW, lub średnia ważona wg funkcji odległości lub atrybutu),
- odchylenie standardowe/wariancja, skośność, kurtozę

- średnie odchylenie bezwzględne (ang. median absolute deviation),
- kwantyle/percentyle (zdefiniowane przez użytkownika),
- mediana,
- moda,
- suma,
- entropia,
- rozstęp (ang. range),
- wartości (interpolacja wysokości, wektor normalny, kierunek największego spadku, nachylenie płaszczyzny) wynikające z wpasowania płaszczyzny w sąsiedztwo, z wykorzystaniem metody najmniejszych kwadratów,
- błąd wpasowania płaszczyzny w sąsiedztwo metodą najmniejszych kwadratów
- odległość pomiędzy danym punktem a płaszczyzną wpasowaną w sąsiedztwo metodą najmniejszych kwadratów.

Algorytm powinien pozwolić na obliczenie w zdefiniowanym sąsiedztwie (z możliwością narzucenia maksymalnej liczby sąsiadów) statystyk dla zdefiniowanego atrybutu/kolumny/kilku kolumn. Algorytm powinien gwarantować możliwości zapisania dla każdego punktu liczby n – z ilu sąsiadów wykonywano obliczenie i wybrania zachowania algorytmu w sytuacji, jeśli n=0 dla jakiegoś puntu. Dla średniej ważonej algorytm powinien umożliwiać wykorzystanie innego atrybutu jako wagi lub na zdefiniowanie własnej funkcji (odległości) wagującej.

3) GETNEIGHBOURS

Algorytm, który oblicza jedną (lub klika) z cech sąsiedztwa, przyjmując jako parametry funkcję wyznaczania sąsiedztwa (np. GetProximity lub własną funkcję użytkownika) i jedną lub klika z funkcji podanych poniżej (lub własne funkcje użytkownika). Zakres obliczeń:

- liczba punktów,
- gęstość na m2,
- gęstość na m3,
- odległość 2D/3D do najbliższego punktu,
- odległość 2D/3D kolejnych n najbliższego punktów.

Algorytm powinien gwarantować możliwości zapisania dla każdego punktu liczby n – z ilu sąsiadów wykonywano obliczenie, pozwalać na określenie górnego limitu liczby punktów i wybrania zachowania algorytmu w sytuacji, jeśli n=0 dla jakiegoś punktu.

4) GETEIGENFEATURES

Algorytm, który dla danego sąsiedztwa liczy wartości własne, przyjmując jako dane definicję sąsiedztwa (z GetProximity) lub strukturę danych zawierającą predefiniowane punkty w sąsiedztwie i jedną lub klika podanych poniżej wartości:

- wartości własne (ang. eigenvalues),
- wektory własne (ang. eigenvectors),
- suma wartości własnych (ang. sum of eigenvalues),
- omniwariancja (ang. omnivariance),
- eigentropia (ang. eigentropy),
- anizotropowość (ang. anisotropy),
- płaszczyznowość (ang. planarity),
- liniowość (ang. linearity),
- wariancja powierzchni (ang. surface variation),

- sferyczność (ang. sphericity),
- pionowość (ang. verticality).

Algorytm powinien pozwalać na obliczenie wartości własnych z dowolnego atrybutu (nie tylko Z).

Algorytm powinien gwarantować możliwości zapisania dla każdego punktu liczby n – z ilu sąsiadów wykonywano obliczenie i wybrania zachowania algorytmu w sytuacji, jeśli n=0 dla jakiegoś puntu.

5) RASTERSTAT

Algorytm pozwalający na utworzenie rastra o zadanej rozdzielczości (obsługiwane przypadki powinny obejmować inną rozdzielczość w kierunkach X i Y oraz zdefiniowanie współrzędnych środka pierwszego piksela). Atrybuty dla poszczególnych węzłów rastra powinny być obliczane z wykorzystaniem metody (lub kilku metod) jak w algorytmach: GetStat, GetMyStat, GetNeighbours, GetEigenFeatures, liczonych w sąsiedztwach od środków pikseli zgodnie z metodami z algorytmu GetProximity.

6) RASTERNEAREST

Algorytm pozwalający na utworzenie rastra o zadanej rozdzielczości (obsługiwane przypadki powinny obejmować inną rozdzielczość w kierunkach X i Y oraz zdefiniowanie współrzędnych środka pierwszego piksela). Atrybuty dla poszczególnych węzłów rastra powinny być przepisane z najbliższego sąsiada w chmurze. Algorytm powinien mieć możliwość określenia maksymalnej odległości poszukiwania najbliższego sąsiada.

7) RASTERNATURAL

Algorytm pozwalający na utworzenie rastra o zadanej rozdzielczości (obsługiwane przypadki powinny obejmować inną rozdzielczość w kierunkach X i Y oraz zdefiniowanie współrzędnych środka pierwszego piksela). Atrybuty dla poszczególnych węzłów rastra powinny być przepisane za pomocą interpolacji naturalnego sąsiedztwa (rozumianego jako https://en.wikipedia.org/wiki/Natural_neighbor_interpolation). Algorytm musi mieć możliwość podania maksymalnej odległości podczas liczenia średniej ważonej.

8) TIN

Algorytm pozwalający na utworzenie sitaki trójkątów (triangulacji w 2D) w oparciu o chmurę punktów i zapisane jej w postaci poligonów lub rastra o zadanej rozdzielczości. Algorytm musi mieć możliwość podania maksymalnej długości boku tworzonego trójkąta. .

9) RASTERIZENODATA

Algorytm służący do interpolacji wartości w rastrze dla tych pikseli, które mają wartość NoData. Do wyboru powinny być następujące metody:

- najbliższego sąsiedztwa,
- metoda naturalnego sąsiedztwa,
- triangulacji,
- wpasowania płaszczyzny,
- średniej,
- średniej ważonej,
- min,
- max.

Wartości są interpolowane z pikseli okalających obszar NoData. Powinno być możliwe zdefiniowanie, maksymalnego obszaru lub odległości interpolacji.

10) GETCLUSTERSTAT

Algorytm pozwalający na wykonanie operacji analogicznych dla tych zdefiniowanych w GetProximity, GetStat, GetNeighbours GetEigenFeatures ale tylko dla punktów należących do klastra. Podział na klastry może być zdefiniowany na trzy sposoby: pochodzić z algorytmu Clusterize, określone dodatkową kolumną (zawierającą identyfikator klastra) lub warstwą poligonową. Statystki mogą być liczone osobno dla każdego punktu w zadeklarowanym sąsiedztwie (analogicznie do tych dostępnych w GetProximity) lub jako jedna statystyka dla wszystkich punktów w klastrze.

11) DECIMATE

Algorytm pozwalający na wybranie (np. poprzez dodanie nowego atrybutu) określonego % punktów z chmury lub zostawanie zadanej ich liczby tak by rozkład wybranych przestrzennych punktów był możliwie równomierny. Wybór powinien mieć charakter dopisania atrybutu.

12) REGULARIZE

Algorytm pozwalający na wybranie (np. poprzez dodanie nowego atrybutu) punktów z chmury tak by zachowana została pomiędzy nimi założoną odległość w 3D lub w 2D. Wybór powinien mieć charakter dopisania atrybutu.

13) LOCALCON

Algorytm pozwalający na wybranie punktów (np. poprzez dodanie nowego atrybutu typu logicznego) z chmury, które w zadanym sąsiedztwie (wg. GetProximity) spełniają warunek geometryczny (dla dowolnego atrybutu) tj. są lokalnym minimum, maksimum lub znajdują się w grupie n największych/najmniejszych wartości w danym sąsiedztwie.

14) CLOUD2CLOUD

Algorytm pozwalający na przenoszenie atrybutów i liczenia statystyk (wg metod z GetStat) pomiędzy dwiema chmurami.

15) DISTANCEC2C

Algorytm pozwalający na obliczanie odległości na podstawie dwóch chmur punktów w następujących wariantach:

- dla każdego punkty z chmury odległość XYZ lub składowe X, Y, Z do najbliższego punktu z drugiej chmury (najbliższego sąsiada możemy szukać w 3D lub 2D).
- dla każdego punkty z chmury odległość do płaszczyzny wpasowaną w lokalne sąsiedztwo w drugiej chmurze.

Algorytm powinien pozwalać na zdefiniowanie maksymalnej odległości.

16) DISTANCEC2V

Algorytm pozwalający na obliczenie odległości (w tym znajdywanie najbliższego w 2D/3D jeśli jest ich wiele) obiektu wektorowego dla każdego punktu w chmurze (obsługa linii lub sitaki trójkątów), dla siatki trójkątów odległość liczona osobno dla XYZ lub wg. normalnej do płaszczyzny trójkąta.

17) LINE2CLOUD

Algorytm pozwalający na zamianę linii 2D lub 3D na punkty (generowanie punktów co zadaną odległość – bieżącą).

18) POLIGON2CLOUD

Algorytm pozwalający na zamianę poligonu 2D na punkty (o określonej rozdzielczości), gdzie Z pochodzi z atrybutu (mogą być też inne atrybuty). Generowanie punktów co zadaną odległość – rozdzielczość.

19) TIN2CLOUD

Algorytm pozwalający na zamianę trójkątów (siatki tin) na punkty, gdzie Z pochodzi z płaszczyzny (mogą być też inne atrybuty). Generowanie punktów co zadaną odległość – rozdzielczość 2D/3D.

20) CLUSTERIZE

Algorytm pozwalający na wykonanie operacji segmentacji/clusteringu chmury punktów. Do wyboru powinny być następujące metody:

- Region Grow,
- Plane Detection,
- K-Means,
- Mean Shift.

Segmentacja powinna działać w 3D, przy czym jako trzecia współrzędna może być wybierany dowolny atrybut (nie tylko z). Algorytmy powinny dawać możliwość definiowania punktów inicjalnych (seed), minimalnej liczby punktów w klastrze/segmencie czy definiowania minimalnej odległości.

21) OUTLINE

Algorytm pozwalający na utworzenie warstwy wektorowej 2D (poligon), której obrys zawiera wszystkie punkty w DataFrame, powinien on obsługiwać przynajmniej dwie metody generowania zasięgu chmury punktów w płaszczyźnie XY: convex hull i alpha-shape.

22) ADDGEORASTER

Algorytm pozwalający na dodanie atrybutu (lub atrybutów np. RGB) dla każdego punktu; wartość atrybutu obliczana jest na podstawie rastra (posiadającego georeferencję).

23) ADDEORASTER

Algorytm pozwalający na dodanie atrybutu (lub atrybutów np. RGB) do każdego punktu; wartość atrybutu obliczana jest na podstawie rastra i znanych elementów orientacji (IO, EO dla zdjęcia kadrowego, RPC dla scen satelitarnych).

24) SVF

Algorytm pozwalający na obliczenie atrybutu „cieniowania” wg. metody Sky-View-Factor dla poszczególnych punktów w chmurze punktów na podstawie analizy zdefiniowanego sąsiedztwa.

25) IO

Biblioteka powinna być wyposażona w funkcje pozwalające na odczyt danych z plików las i laz, txt (wg. zdefiniowanego przez użytkownika schematu), ply, CloudCompare bin oraz potree do Pandas

Dataframe. Konwersja danych z Pandas Dataframe do Spark DataFrame nie wchodzi w zakres opracowania funkcji.

Biblioteka powinna być wyposażona w funkcje pozwalające na zapis danych z Pandas DataFrame do plików las i laz, txt (wg. zdefiniowanego przez użytkownika schematu), ply, CloudCompare bin, potree. Konwersja danych z Spark Dataframe do Pandas DataFrame i podział danych wejściowych na podzbiory nie wchodzi w zakres opracowania funkcji.

Zarówno odczyt jak i zapis danych powinien wspierać odczyt/zapis z/do systemu plików HDFS.

B. OPROGRAMOWANIE DO WIZUALIZACJI I ANALIZY CHMUR PUNKTÓW

Oprogramowanie typu desktop, przeznaczone do wizualizacji chmur punktów powinno działać co najmniej w środowisku Windows (64 bitowym) na maszynach wirtualnych z Windows 10. Oprogramowanie powinno zostać dostarczone w liczbie minimum 10 licencji (umożliwiających jednoczesne uruchomienie na minimum 10 maszynach) pływających (pobierane z lokalnego serwera licencji).

WYMAGANIA OGÓLNE

Dostarczone oprogramowanie powinno spełniać co najmniej poniższe wymagania:

1) Co najmniej trzy tryby wyświetlania: ortogonalny, perspektywiczny (możliwość definicji kamery przez dwa punkty i ogniskową) i przekrój (w tym możliwość definicji przekroju wzdłuż łamanej o zadanej długości i szerokości).
2) Wyświetlanie chmur punktów (kolorowanych wg atrybutów RGB, Intensywność, Klasyfikacja, palety barwnej związanej z wysokością oraz umożliwiać zapis/odczyt schematów wyświetlania).
3) Wyświetlanie danych rastrowych (w trybie ortogonalnym).
4) Wyświetlanie modelu TIN tworzonego (w locie) na podstawie chmur punktów.
5) Wyświetlanie modeli 3D w postaci: siatki trójkątów, jednolitej powierzchni, powierzchni z teksturą.
6) Odczyt następujących formatów danych dla chmur punktów z plików: TXT, LAS i LAZ i PTS; dla danych rastrowych (JPEG, TIFF, ECW); dla modeli 3D: GML, OBJ, FBX, IFC.
7) Odczyt danych z WMS (w trybie ortogonalnym).
8) Możliwość pomiaru odległości i powierzchni.
9) Tworzenie map gęstości z wczytanych chmur punktów.
10) Możliwość manualnej klasyfikacji chmury punktów (wewnątrz prostokąta, poligonu, powyżej linii, poniżej linii)
11) Wykorzystanie GPU w celu akceleracji wyświetlania oraz operacji obliczeniowych aplikacji.
12) Spójność wyników realizowanych obliczeń z wynikami obliczeń wykonywanymi w ramach analogicznych funkcji biblioteki do obliczeń na chmurach punktów i danych rastrowych opisanej w punkcie 2.A na stronie 1.

LICENCJA

Oprogramowanie typu desktop do wizualizacji i analizy chmur punktów dostarczone w ramach zamówienia jest oprogramowaniem gotowym, na które Wykonawca udzieli licencji niewyłącznej, na czas nieokreślony i nieograniczonej terytorialnie w zakresie koniecznym dla wykonania i korzystania niniejszego Zamówienia.

Na podstawie udzielonej licencji Zamawiający uprawniony będzie do korzystania z oprogramowania na polach eksploatacji wymienionych poniżej:

a) użytkowanie w zakresie wynikającym z jego charakteru i przeznaczenia,
b) trwałego lub czasowego zwielokrotnienia (sporządzenia kopii) lub utrwalania całości lub części do ilości niezbędnej dla celów bezpiecznej i efektywnej eksploatacji, instalowania na komputerach posiadanych przez Zamawiającego,
c) korzystanie w celu przetwarzania wszelkich danych przez użytkowników platformy CENAGIS, w tym przez pracowników Zamawiającego, osób działających na zlecenie Zamawiającego, klientów Zamawiającego.

Licencja musi uwzględniać prawo do bezpłatnej instalacji udostępnianych przez producenta uaktualnień i poprawek krytycznych i opcjonalnych w okresie gwarancji.

3. ETAPY ZAMÓWIENIA

Realizacja niniejszego zamówienia została podzielona na następujące etapy:

ETAP I – ANALIZA WYMAGAŃ

W ramach Etapu I Wykonawca zidentyfikuje szczegółowe potrzeby i doprecyzuje wymagania Zamawiającego. Następnie Wykonawca opracuje projekt techniczny i scenariusze testowe biblioteki.

Projekt będzie zawierał przynajmniej charakterystykę rozwiązania, opis proponowanej technologii wykonania, szczegółowy opis funkcjonalności, interfejsów, struktur danych, konfiguracji, instalacji, metod i przykładów wykorzystania.

Projekt techniczny oraz scenariusze testowe biblioteki będą podlegały akceptacji przez Zamawiającego.

ETAP II – OPRACOWANIE I DOSTAWA BIBLIOTEKI DO OBLICZEŃ NA CHMURACH PUNKTÓW I DANYCH RASTROWYCH

Na podstawie stworzonego we wcześniejszym etapie projektu technicznego, Wykonawca wykona oraz dostarczy bibliotekę do obliczeń na chmurach punktów i danych rastrowych w wersji produkcyjnej. Wykonawca, wraz z biblioteką dostarczy dokumentacje użytkownika zawierającą przynajmniej opis działania i przykłady zastosowania poszczególnych funkcji biblioteki.

W ramach niniejszego etapu przeprowadzone zostaną testy akceptacyjne biblioteki w wersji produkcyjnej w oparciu o przygotowane scenariusze testowe.

ETAP III – DOSTAWA OPROGRAMOWANIA TYPU DESKTOP DO WIZUALIZACJI I ANALIZY CHMUR PUNKTÓW.

W ramach niniejszego etapu Wykonawca dostarczy oprogramowanie typu desktop do wizualizacji i analizy chmur punktów wraz dokumentacją użytkownika i administratora.

ETAP IV – ODBIORY

W oparciu o wyniki testów zrealizowanych w ramach Etapu II dokonane zostaną formalne odbiory dostarczonych produktów. Etap IV zakończy się sporządzeniem protokołu odbioru.

4. HARMONOGRAM ZAMÓWIENIA

Wymaga się, aby Wykonawca opracował oraz przedstawił szczegółowy harmonogram zamówienia w ciągu 5 dni od podpisania umowy. Harmonogram stanowi narzędzie kontroli i monitorowania postępu prac. Opracowany Harmonogram musi być spójny i musi zawierać proponowane terminy realizacji poszczególnych etapów oraz zadań wchodzących w zakres prac w ramach etapów.

Harmonogram musi uwzględniać czas potrzebny Zamawiającemu na weryfikację dostarczonych produktów oraz czas potrzebny Wykonawcy na uwzględnienie wyników tej weryfikacji.

5. GWARANCJA

Wykonawca udzieli minimum 36 miesięcznej gwarancji na poprawną pracę dostarczonych produktów. Ostateczny okres gwarancji stanowi jedno z kryteriów oceny ofert w SIWZ. Gwarancja obejmuje wszystkie produkty, dostawy i usługi objęte niniejszym postępowaniem oraz modyfikacje, aktualizacje systemu realizowane przez Wykonawcę w ramach gwarancji i wsparcia technicznego, oraz modyfikacje wykonane przez Zamawiającego o ile zostały wcześniej autoryzowane (wyrażona zgoda) przez Wykonawcę.

Termin rozpoczęcia gwarancji liczony jest od dnia podpisania protokołu odbioru końcowego bez uwag.

Gwarancja świadczona będzie od poniedziałku do piątku w godzinach 8.00-16.00.

W ramach gwarancji Wykonawca zobowiązuje się bezpłatnie do:

- analizy i usuwania wykrytych usterek oraz błędów,
- usuwania przyczyn oraz skutków usterek i błędów,
- dostarczania, instalacji (po uzyskaniu zgody Zamawiającego) i konfiguracji łatek i poprawek dla dostarczonych produktów oraz związanych z tym aktualizacji dostarczonej dokumentacji i przekazanych kodów źródłowych,

Czas reakcji Wykonawcy na otrzymane zgłoszenie nie może być dłuższy niż 2 godziny. (Brak potwierdzenia we wskazanym czasie oznacza, iż zgłoszenie zostało przyjęte i po upływie 2 godzin od momentu zgłoszenia rozpoczyna się bieg terminu skutecznej naprawy).

Wykonawca niezwłocznie po otrzymaniu zgłoszenia przystąpi do analizy zaistniałej sytuacji i podejmie działania zmierzające do usunięcia zgłoszonych nieprawidłowości w działaniu produktów.

Czasy realizacji zgłoszeń:

- czas skutecznej naprawy błędu krytycznego (stanu oprogramowania lub jego części, w którym niemożliwa jest jego eksploatacja) to maksymalnie 12 godzin od momentu potwierdzenia przyjęcia zgłoszenia.
- czas skutecznej naprawy błędu (stanu oprogramowania lub jego części, który powoduje ze daje on błędne lub niezgodne z dokumentacją wyniki działania) to maksymalnie 48 godzin od momentu potwierdzenia przyjęcia zgłoszenia.
- czas skutecznej naprawy usterki (działania oprogramowania lub jego części, niezgodnego z dokumentacją, które nie wpływa w sposób istotny na wyniki jego działania lub sposób jego eksploatacji) to maksymalnie 96 godzin od momentu potwierdzenia przyjęcia zgłoszenia.

Z każdej czynności realizowanej w ramach gwarancji (instalacja, konfiguracja, modyfikacja, itp.) Wykonawca sporządzi protokół zawierający m.in. powód przeprowadzonych prac, ich szczegółowy zakres oraz czas rozpoczęcia i zakończenia prac. Upoważniony pracownik Zamawiającego potwierdza wykonanie prac podpisują przygotowany przez Wykonawcę protokół. Zamawiający dopuszcza możliwość realizacji tego wymagania w ramach dedykowanego narzędzia do zgłaszania i śledzenia błędów, które na czas i w ramach realizacji umowy udostępni Wykonawca.

Termin każdej instalacji musi zostać uzgodniony z Zamawiający minimum na 3 dni robocze przed planowanym rozpoczęciem prac (termin nie dotyczy czynności wynikającej z naprawy błędu).

6. WSPARCIE TECHNICZNE

Wykonawca zapewni pulę 300 roboczo-osobo-godzin, do wykorzystania w okresie 60 miesięcy od dnia podpisania protokołu odbioru końcowego bez uwag, w ramach których Zamawiający może zlecać Wykonawcy wykonanie prac związanych z uzupełnianiem, usprawnianiem, przystosowaniem systemu do jego rozbudowy i rozwoju lub jego modyfikacją.

Zamawiający będzie zlecał wykonawcy pracę według własnych potrzeb. Sposób ich realizacji, harmonogram i wycena godzinowa prac będą przedmiotem szczegółowych ustaleń pomiędzy Zamawiającym a Wykonawcą poczynionych na etapie specyfikowania i analizy zlecenia.