Szczegółowy opis przedmiotu zamówienia

1.1. Opis przedmiotu zamówienia

Przedmiotem zamówienia jest wykonanie usług informatycznych w zakresie projektu „e-Puszcza. Podlaskie cyfrowe repozytorium przyrodniczych danych naukowych", polegających na aktualizacji komponentów bazowych oraz rozszerzeniu funkcjonalności bazo-danowej aplikacji internetowej do zarządzania projektami foto-pułapkowymi TRAPPER (https://gitlab.com/oscf/trapper-project) wraz z opracowaniem zestawu testów (unit tests oraz testy REST API) i pełnej dokumentacji technicznej aplikacji oraz z asystą techniczną w wymiarze 100 godzin w okresie 6 miesięcy liczonych od daty odbioru prac. Po zakończeniu prac **aplikacja musi zachować dotychczasową funkcjonalność rozbudowaną o funkcjonalność powstałą w ramach realizacji zadań**.

1.2. Opis aplikacji TRAPPER

TRAPPER jest bazo-danową, interaktywną aplikacją sieciową open-source wspierającą zarządzanie projektami foto-pułapkowymi, klasyfikację nagrań oraz współdzielenie i ponowne wykorzystanie zebranych danych. TRAPPER będzie wykorzystywany w projekcie POPC „e-Puszcza. Podlaskie cyfrowe repozytorium przyrodniczych danych naukowych" do organizacji, zarządzania i generowania informacji naukowej na podstawie zdjęć i filmów zwierząt zarejestrowanych przez foto-pułapki w zrealizowanych i przyszłych projektach badawczych i monitoringowych z wykorzystaniem foto-pułapek. Kod źródłowy aplikacji TRAPPER, dostępny na otwartej licencji GNU GPL v.3, znajduje się w otwartym repozytorium gitlab pod adresem: https://gitlab.com/oscf/trapper-project. Kod źródłowy, na bazie którego Wykonawca zrealizuje wymienione poniżej zadania, znajduje się w branchu *development*. Dokumentacja aktualnej wersji aplikacji TRAPPER znajduje się pod adresem: https://trapper-project.readthedocs.io/en/latest/index.html. Opis systemu wraz z załączonymi wizualizacjami modelu oraz przepływu danych w systemie TRAPPER opublikowany został w czasopiśmie

Methods in Ecology and Evolution i jest dostępny nieodpłatnie pod adresem: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/2041-210X.12571.

1.3. W ramach zamówienia będą realizowane prace w następującym zakresie:

(a) Analizy, projektowania, wytworzenia i modyfikacji aplikacji oraz integracji z zewnętrznymi systemami Zamawiającego;

(b) Wsparcia w instalacji oraz konfiguracji aplikacji i systemów teleinformatycznych, o których mowa w postępowaniu na środowisku będącym w posiadaniu Zamawiającego;

(c) Wytwarzania i aktualizacji dokumentacji technicznej aplikacji;

(d) Usuwania błędów w istniejących i modyfikowanych aplikacjach;

(e) Testów zmodyfikowanych aplikacji i systemów teleinformatycznych;

(f) Wsparciu Zamawiającego w obsłudze zgłoszeń użytkowników;

(g) Wsparciu Zamawiającego w zakresie bieżącej obsługi związanej z realizacją i rozliczaniem prac projektowych

1.4. W ramach zamówienia opisanego powyżej przedmiotu zamówienia zostaną zrealizowane następujące zadania:

1.4.1. Aktualizacja komponentów bazowych aplikacji TRAPPER

- aktualizacja części kodu źródłowego aplikacji napisanego w języku Python do wersji Python 3.6+, Django 2.2+ oraz Django Rest Framework 3.9+ lub nowszych, stabilnych,
- aktualizacja wszystkich komponentów aplikacji i kodu źródłowego w celu zapewnienia kompatybilności z najnowszymi stabilnymi wersjami nginx, RabbitMQ, PostgreSQL, PostGIS oraz Celery,
- aktualizacja wszystkich komponentów frontend'u, w tym m.in. Angular.js i Boostrap do najnowszych stabilnych wersji; lista wszystkich bibliotek, które zostaną zaktualizowane przez Wykonawcę znajduje się pod tym adresem:

  https://trapper-project.readthedocs.io/en/latest/technologies.html#frontend

- aktualizacja komponentu mapowego aplikacji TRAPPER (aplikacja django *geomap*) w celu zapewnienia kompatybilności zarówno części backend'owej jak i frontend'owej z najnowszą wersją aplikacji webGIS'owej UMAP (https://github.com/umap-project/umap),
- aktualizacja wszystkich pozostałych dodatkowych pakietów zewnętrznych (3rd-party) Python i Django w celu zapewnienia kompatybilności z wyżej wymienionymi aktualizacjami,
- po wykonaniu wszystkich wymienionych powyżej aktualizacji aplikacja TRAPPER musi zachować dotychczasową funkcjonalność,

1.4.2. Analiza kodu źródłowego po aktualizacjach i refaktoryzacja w celu optymalizacji wydajności

- wykonanie testów obciążeniowych tj. zestawienie środowiska testowego oraz wykonanie testów,
- środowisko do testów obciążeniowych zostanie dostarczone przez Wykonawcę w postaci kodu źródłowego umożliwiającego uruchomienie testów na dowolnym serwerze z działającą instancją aplikacji TRAPPER,
- przygotowanie raportu z analiz wydajności aplikacji,
- optymalizacja jakości i organizacji kodu oraz optymalizacja wydajności aplikacji,
- analizie i refaktoryzacji poddany zostanie kod źródłowy związany z filtrowaniem obiektów w bazie danych (*querysets*) i systemem kontroli dostępu do zasobów (*user permissions & roles*),
- gdziekolwiek będzie to możliwe preferowane jest upraszczanie kodu i usuwanie jego zbędnych fragmentów,

Testy obciążeniowe wykonane zostaną przy uwzględnieniu następujących parametrów:

- min. 20 milionów zasobów (model *geomap.Resource*) zarejestrowanych na min. 100 tysiącach lokalizacji (model *storage.Location*) podczas min. 200 tysięcy deploymentów (model *geomap.Deployment*),
- max. ilość zasobów zarejestrowana podczas jednego deploymentu to 200,

- max. ilość deploymentów na jednej lokalizacji to 4,
- zasoby zorganizowane w min. 10 tysięcy kolekcji (*model storage.Collection*), przy czym jeden zasób może należeć do wielu kolekcji,
- min. 10 tysięcy użytkowników, przy czym każdy użytkownik ma dostęp do min. 100 tysięcy zasobów i min. 500 kolekcji,
- min. 100 zdefiniowanych projektów badawczych (model *research.ResearchProject*), przy czym każdy projekt badawczy zawiera min. 1 milion zasobów, min. 10 użytkowników w roli *Admin*, min. 1 tysiąc użytkowników w roli *Expert* oraz min. 1 tysiąc użytkowników w roli *Collaborator,*
- 2 zdefiniowane projekty klasyfikacyjne (model *media_classification.ClassificationProject*) na każdy projekt badawczy, po pół miliona zasobów przypisanych do każdego projektu klasyfikacyjnego; min. 10 użytkowników w roli *Admin*; pozostali użytkownicy powiązanego projektu badawczego mają rolę *Collaborator* w obu projektach klasyfikacyjnych oraz min. 2 miliony klasyfikacji (modele *media_classification.UserClassification* oraz *media_classification.Classification*) w ramach każdego z projektów,
- konfiguracja serwera do testów: 8 GB RAM Intel(R) Xeon(R) CPU E5-2430 v2 @ 2.50GHz (4 cores); serwer VPS do testów zostanie udostępniony Wykonawcy przez Zamawiającego,
- testy zostaną przeprowadzone dla 1000 użytkowników i dla każdej z 4 kategorii uprawnień: *Superuser*, *Admin*, *Expert* i *Collaborator* (w sumie cztery tysiące wywołań bazy danych dla danego testu),

Oczekiwane średnie czasy reakcji dla poszczególnych wywołań API podczas testów z wymienionymi powyżej parametrami:

- **storage/api/resources** - max. 500ms,
- **geomap/api/locations** - max. 200ms,
- **geomap/api/deployments** - max. 200ms,
- **media_classification/api/classifications?project=X** - max. 300ms.

1.4.3. Wdrożenie systemu wielojęzyczności w aplikacji TRAPPER

- wykorzystanie mechanizmów dostępnych w django:

  https://docs.djangoproject.com/en/2.2/topics/i18n/translation/,

- aktualizacja wszystkich fragmentów kodu źródłowego generującego informacje wyświetlane i wysyłane użytkownikom,
- tłumaczenie treści na inne języki nie jest częścią tego zadania.

1.4.4. Aktualizacja istniejących i implementacja brakujących testów jednostkowych kodu źródłowego aplikacji TRAPPER

- aktualizacja istniejących i implementacja brakujących testów jednostkowych (unit tests) funkcjonalności backendu (django) oraz testów REST API.
- oczekiwane pokrycie testami będzie nie mniejsze niż 80%, mierzone narzędziem „Coverage.py” (https://coverage.readthedocs.io/en/v4.5.x/)
- testami jednostkowymi oraz testami REST API pokryta zostanie funkcjonalność następujących modułów django aplikacji TRAPPER:
  - accounts,
  - storage,
  - research,
  - geomap,
  - media_clasification,
  - sendfile,
  - messaging,
  - common.

1.4.5. Opracowanie pełnej technicznej dokumentacji aplikacji TRAPPER

- uzupełnienie technicznej dokumentacji wszystkich klas i funkcji w kodzie źródłowym,

- dokumentacja kodu źródłowego w języku Python wykonana zostanie w formacie reStructuredText (reST),
- opracowanie pełnej dokumentacji technicznej w środowisku Sphinx (http://www.sphinx-doc.org/en/master/),
- dokumentacja techniczna wykonana zostanie w języku angielskim.

1.4.6. Implementacja uniwersalnego mechanizmu eksportu wyników projektów klasyfikacyjnych

- implementacja logiki uniwersalnego mechanizmu eksportu wyników projektów klasyfikacyjnych w wielu formatach (opisane poniżej),
- zadanie dotyczy modelu *Classification* w aplikacji django *media_classification* oraz powiązanych modeli, w tym *ClassificationProject* oraz *Classificator*, a także modeli *Deployment* oraz *Location* w aplikacji django *geomap*,
- w aktualnej wersji aplikacji TRAPPER istnieje mechanizm eksportu wyników projektu klasyfikacyjnego do pliku zip zawierającego tabele z klasyfikacjami i deploymentami oraz dodatkowy plik z metadanymi w formacie EML (Ecologcial Metadata Language),
- mechanizm ten zaimplementowany jest w widoku:

  https://$host/media_classification/classification/export/
- dodatkowo istnieje API generujące tabele z klasyfikacjami oraz deploymentami:

  https://$host/media_classification/api/classifications/results/$pk/

  https://$host/geomap/api/deployments/export/

Wykonawca uogólni i rozszerzy ten mechanizm aktualizując istniejące i implementując nowe punkty końcowe API oraz aktualizując istniejące widoki.

Metadane będą generowane zarówno na poziomie opisującym sam projekt klasyfikacyjny (*ClassificationProject*) oraz nadrzędny projekt badawczy (*ResearchProject*), w ramach którego klasyfikacje zostały wykonane, jak i na poziomie opisującym strukturę samych obiektów klasyfikacji i deploymentów, w tym lista i opis atrybutów klasyfikacji.

Wykonawca zaimplementuje mechanizm eksportu metadanych w następujących formatach:

- EML (Ecological Metadata language)

  https://knb.ecoinformatics.org/external//emlparser/docs/index.html (funkcjonalność częściowo zaimplementowana w aktualnej wersji aplikacji TRAPPER)

- Dublin Core / Darwin Core

  https://dwc.tdwg.org/terms/

- Open Camera Trap Metadata Standard

  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5267527/

- Zenodo

  https://zenodo.org/schemas/records/record-v1.0.0.json

- DATAVERSE

  http://guides.dataverse.org/en/latest/api/native-api.html

Wykonawca umożliwi eksport wyników klasyfikacji (model *media_classification.Classification*) i powiązanych z nimi deploymentów (model *geomap.Deployment*) i lokalizacji (model *geomap.Location*) w następujących formatach:

- Natywnym dla aplikacji TRAPPER (zaimplementowany w aktualnej wersji)

  https://$host/media_classification/api/classifications/results/$pk/

- camtrapR

  https://cran.r-project.org/web/packages/camtrapR/index.html

- Wildlife Insights

  https://github.com/ConservationInternational/Wildlife-Insights----Data-Migration
  https://docs.google.com/spreadsheets/d/1PE5Zl-HUG4Zt0PwSfj-gJRJVbZ__LgH3VuiDW3-BKQg/edit#gid=1190405668

- DATAVERSE

  http://guides.dataverse.org/en/latest/admin/metadatacustomization.html
  http://guides.dataverse.org/en/latest/api/native-api.html

Eksport do opisanych powyżej formatów metadanych oraz wyników klasyfikacji i powiązanych z nimi deploymentów i lokalizacji zaimplementowany zostanie w postacie nowych lub

zaktualizowanych punktów końcowych API. Dodatkowo zaktualizowany zostanie widok https://$host/media_classification/classification/export/, który będzie umożliwiać użytkownikowi wybór formatów plików, które znajdą się w wygenerowanej paczce zip z danymi.

1.4.7. Integracja aplikacji TRAPPER z repozytorium danych DATAVERSE

- informacje i dokumentacja systemu DATAVERSE znajduje się pod adresem:

  https://dataverse.org/

  http://guides.dataverse.org/en/latest/api/client-libraries.html#id1,

- Wykonawca zaimplementuje logikę oraz widoki eksportu wyników projektów klasyfikacyjnych do repozytorium DATAVERSE bezpośrednio z poziomu aplikacji TRAPPER,
- w szczególności Wykonawca zaimplementuje logikę tworzenia z poziomu aplikacji TRAPPER takich obiektów DATAVERSE jak *dataverse*, *dataset* oraz *datafile* wraz z powiązanymi z nimi metadanymi opisującymi eksportowany projekt badawczy i klasyfikacyjny oraz tabele z danymi (klasyfikacje, deploymenty i lokalizacje),
- rozszerzona zostanie aplikacja django *accounts* w celu umożliwienia użytkownikowi aplikacji TRAPPER, w widoku profilu (https://$host/accounts/profile/), konfiguracji korespondującego konta w DATAVERSE tj. takich parametrów jak *host, token, username* oraz innych niezbędnych do prawidłowego działania implementowanej funkcjonalności,
- Wykonawca zaimplementuje mechanizm referencji do fizycznych zasobów (tj. filmów i zdjęć z foto-pułapek) przechowywanych w systemie TRAPPER bezpośrednio z poziomu DATAVERSE, który umożliwiać będzie autoryzowany dostęp do kolekcji multimediów powiązanych z danymi i metadanymi zarchiwizowanymi w repozytorium DATAVERSE.

1.4.8. Integracja aplikacji TRAPPER z logowaniem po protokole LDAP

- Wykonawca zaimplementuje mechanizm uwierzytelniania się do systemu TRAPPER przy wykorzystaniu zewnętrznego serwera implementującego protokół LDAP. W pliku konfiguracyjnym musi znaleźć się możliwość włączenia tego trybu, jak również adresu serwera LDAP, portu, filtru wyszukiwania i atrybutu po którym aplikacja ma odpytywać

serwer LDAP. Po włączeniu tej funkcjonalności, domyślnie aplikacja TRAPPER w pierwszej kolejności będzie uwierzytelniała użytkowników w oparciu o serwer LDAP, następnie w oparciu o lokalną bazę SQL.

- Wykonawca zaimplementuje możliwość wyłączenia panelu rejestracji użytkowników w systemie TRAPPER.
- Wykonawca wykona testy funkcjonalne zaimplementowanej funkcjonalności uwierzytelniania minimum z serwerami OpenLDAP i ActiveDirectory wskazanymi przez Zamawiającego.
- Wykonawca zaimplementuje logikę, zgodnie z którą po poprawnym uwierzytelnieniu użytkownika w bazie LDAP i przy jednoczesnym jego braku w bazie lokalnej SQL, użytkownik zostanie stworzony w tej lokalnej bazie z domyślnymi najmniejszymi uprawnieniami bez zapisania jego hasła.
- Mechanizm integracji z serwerami LDAP służy jedynie do możliwości uwierzytelniania użytkowników, zarządzanie uprawnieniami i pozostałe funkcjonalności pozostają jak dotychczas bez zmian i będą wykonywane bezpośrednio w aplikacji TRAPPER.

1.4.9. Integracja aplikacji TRAPPER z mechanizmem SSO zgodnej ze standardem SAML 2.0

- Wykonawca zaimplementuje mechanizm uwierzytelniania się do systemu TRAPPER przy wykorzystaniu zewnętrznego serwera dostawcy tożsamości Identity Provider (IdP). W pliku konfiguracyjnym musi znaleźć się możliwość włączenia tego trybu, jak również zdefiniowania Entity ID oraz wymaganych atrybutów. Po włączeniu tej funkcjonalności, na stronie logowania aplikacji TRAPPER powinna pojawić się możliwość wyboru jednego ze zdefiniowanych dostawców tożsamości IdP, po jego wyborze ma nastąpić przekierowanie na stronę dostawcy tożsamości, który po poprawnym uwierzytelnieniu i zatwierdzeniu przez użytkownika chęci przekazania wymaganych atrybutów zwróci je do aplikacji TRAPPER.
- Wykonawca zaimplementuje możliwość wyłączenia panelu rejestracji użytkowników w systemie TRAPPER.
- Wykonawca wykona testy funkcjonalne zaimplementowanej funkcjonalności uwierzytelniania minimum z dostawcami tożsamości opartymi o oprogramowanie Shibboleth oraz SimpleSAMLPhp wskazanymi przez Zamawiającego.

- Wykonawca zaimplementuje logikę, zgodnie z którą po poprawnym uwierzytelnieniu użytkownika przez dostawcę tożsamości IdP i przy jednoczesnym jego braku w bazie lokalnej SQL, użytkownik zostanie stworzony w tej lokalnej bazie z domyślnymi najmniejszymi uprawnieniami oraz z przekazanymi atrybutami wymaganymi oraz opcjonalnymi (zdefiniowanymi wcześniej w pliku konfiguracyjnym), jak imię, nazwisko, adres email itp. Dodatkowo po każdym logowaniu poprzez IdP, w przypadku zmiany tych atrybutów należy zaktualizować lokalną bazę SQL.
- Mechanizm integracji z mechanizmem SSO zgodnym ze standardem SAML 2.0 służy jedynie do możliwości delegowania uwierzytelniania innym instytucjom, zarządzanie uprawnieniami i pozostałe funkcjonalności pozostają jak dotychczas bez zmian i będą wykonywane bezpośrednio w aplikacji TRAPPER.

1.5. Wymagania - styl kodowania

- styl kodowania obowiązujący Wykonawcę musi być zgodny ze standardem pep8 (https://www.python.org/dev/peps/pep-0008/) dla wszystkich fragmentów kodu źródłowego w języku Python,
- oryginalne fragmenty kodu źródłowego aplikacji TRAPPER w języku Python, które nie spełniają standardu pep8 zostaną poprawione przez Wykonawcę w celu uzyskania zgodności z tym standardem,
- jeśli dotychczasową funkcjonalność systemu można po aktualizacji komponentów bazowych zrealizować bez pomocy zewnętrznych, dodatkowych bibliotek i/lub pakietów tj. funkcjonalność taka jest zaimplementowana w nowych wersjach komponentów bazowych, Wykonawca będzie wykorzystywał takie możliwości; przykładem może być typ pola w bazie danych Postgresql Hstore, który jest dostępny natywnie w wersji Django 2.2, nie ma więc potrzeby wykorzystania zewnętrznej aplikacji django django-hstore (https://github.com/djangonauts/django-hstore), która zapewnia taką funkcjonalność w aktualnej wersji systemu TRAPPER,
- wszystkie klasy i funkcje w kodzie źródłowym aplikacji TRAPPER muszą mieć opis dokumentujący ich działanie i/lub przeznaczenie oraz wymagane argumenty i zwracane wartości (w przypadku funkcji),
- dokumentacja klas i funkcji w kodzie źródłowym w języku Python musi być w formacie

reStructuredText (reST) umożliwiającym automatyczne wygenerowanie dokumentacji w systemie Sphinx http://www.sphinx-doc.org/.

1.6. Pozostałe wymagania

1.6.1. Przedmiot zamówienia zostanie testowo uruchomiony na platformie sprzętowej (serwerach) Zamawiającego.

1.6.2. Wykonawca zagwarantuje, iż przedmiot zamówienia będzie:

- wolny od wad prawnych i roszczeń osób trzecich;
- wolny od mechanizmów blokujących jego funkcjonowanie.

1.6.3. Zamawiający wymaga udzielenia gwarancji na oferowany przedmiot zamówienia na warunkach jak niżej:

- gwarancja min. 6 miesięcy, rozumiana jako naprawa błędów w funkcjonowaniu aplikacji TRAPPER wynikających z błędnej implementacji zadań realizowanych przez Wykonawcę w zakresie przedmiotu zamówienia,
- gwarancja liczona jest od daty podpisania końcowego protokołu odbioru przez zamawiającego.

1.6.4. Wymagane warunki świadczenia serwisu gwarancyjnego:

- czas reakcji serwisu gwarancyjnego w ciągu max. 1 dnia roboczego od momentu zgłoszenia błędu drogą mailową lub w postaci issues definiowanych w repozytorium kodu gitlab wskazanym przez Zamawiającego,
- czas naprawy od momentu zgłoszenia w ciągu max. 3 do 7 dni roboczych, w zależności od stopnia skomplikowania poprawki.

1.6.5. Wymagania w zakresie sposobu wykonywania Zlecenia:

Zamawiający wymaga, aby Wykonawca realizował powierzone zadania z uwzględnieniem zasad zwinnych metodyk wytwórczych, w szczególności:

- Zadania określone w Zleceniu są realizowane w sposób iteracyjny i przyrostowy w ramach Sprintów o wskazanej w postępowaniu długości (Sprint może trwać od 1 do 4 tygodni).

- Każdy Sprint rozpoczyna się od ustalonego zakresu funkcjonalności lub prac do realizacji w danym Sprincie.
- W przypadkach uzgodnionych z Zamawiającym i za jego zgodą określona część prac może przejść na kolejny Sprint.
- Każdy Sprint kończy się dostarczeniem Zamawiającemu przez Wykonawcę produktu: funkcjonalności lub innego wyniku prac, którym może być gotowa konfiguracja, dokument, wynik testów itp., w zależności od przypisanego Zadania.
- Zadania realizowane w ramach Zlecenia będą podlegały określonym w Umowie procedurom odbiorczym.
- Wykonawca będzie korzystał z oficjalnego repozytorium projektu TRAPPER znajdującego się pod adresem https://gitlab.com/oscf/trapper-project, w którym będzie realizował kolejne Sprinty związane z implementacją poszczególnych zadań w dedykowanych branch'ach. Wykonawca będzie systematycznie wysyłał zmiany w kodzie do powyższego repozytorium w celu archiwizacji prac.

1.7. Wymagania w zakresie tworzenia i aktualizacji dokumentacji

Cała dokumentacja techniczna powinna być opracowywana w języku angielskim. Zamawiający wymaga, aby dokumenty tworzone i modyfikowane w ramach realizacji Zlecenia charakteryzowały się wysoką jakością, właściwą dla profesjonalnego charakteru świadczonych usług, na którą będą miały wpływ, takie czynniki jak:

- struktura dokumentu, rozumiana jako podział danego dokumentu na rozdziały, podrozdziały i sekcje w czytelny i zrozumiały sposób;
- sposób pisania rozumiany jako zachowanie spójnej struktury, formy i sposobu pisania dla poszczególnych dokumentów oraz fragmentów tego samego dokumentu;
- poprawność ortograficzna, gramatyczna i stylistyczna dokumentów;
- utrzymywanie aktualnych powiązań z innymi dokumentami;
- kompletność dokumentu – pełne przedstawienie omawianego problemu obejmujące całość z danego zakresu rozpatrywanego zagadnienia;
- spójność i niesprzeczność dokumentu – zapewnienie wzajemnej zgodności pomiędzy

wszystkimi rodzajami informacji umieszczonymi w dokumencie, brak logicznych sprzeczności pomiędzy informacjami zawartymi we wszystkich przekazanych dokumentach oraz fragmentach tego samego dokumentu;

- aktualność – uwzględnienie w dokumencie bieżących czynników i uwarunkowań, w tym aktualnie istniejącej dokumentacji;
- zachowanie ogólnie przyjętych norm, standardów i kryteriów jakości;

1.8. Czas realizacji oraz etapy realizacji poszczególnych zadań

Czas realizacji całego zamówienia to max. pięć miesięcyce od momentu podpisania umowy. Pojedyncze zadania będą realizowane w formie opisanych w pkt. 1.6.5 Sprintów. Jednocześnie, Zamówienie będzie realizowane w następujących etapach:

Etap 1:

- zadanie 1.4.1: *Aktualizacja komponentów bazowych aplikacji TRAPPER*
- zadanie 1.4.2: *Aktualizacja istniejących i implementacja brakujących testów jednostkowych kodu źródłowego aplikacji TRAPPER*
- czas realizacji: max. 2 miesiące od podpisania umowy

Etap 2:

- zadanie 1.4.2: *Analiza kodu źródłowego po aktualizacjach i refaktoryzacja w celu optymalizacji wydajności*
- zadanie 1.4.3: *Wdrożenie systemu wielojęzyczności w aplikacji TRAPPER*
- czas realizacji: max. 1 miesiąc od zakończenia Etapu 1

Etap 3:

- zadanie 1.4.6: *Implementacja uniwersalnego mechanizmu eksportu wyników projektów klasyfikacyjnych*
- zadanie 1.4.7: *Integracja aplikacji TRAPPER z repozytorium danych DATAVERSE*
- czas realizacji: max. 1 miesiąc od zakończenia Etapu 2

Etap 4:

- zadanie 1.4.8: *Integracja aplikacji TRAPPER z logowaniem po protokole LDAP*
- zadanie 1.4.9: *Integracja aplikacji TRAPPER z mechanizmem SSO zgodnej ze standardem SAML 2.0*
- czas realizacji: max. 1 miesiąc od zakończenia Etapu 3

Zadanie 1.4.5: *Opracowanie pełnej technicznej dokumentacji aplikacji TRAPPER* realizowane będzie równolegle do realizacji pozostałych zadań.