

## 5 Biblioteka X2M

Biblioteka X2M, będąca rozwiązańem postawionego zadania w niniejszej pracy magisterskiej, zaimplementowana została w postaci API (ang. Application programming interface) – interfejsu programistycznego aplikacji, który należy rozumieć jako zestaw reguł, struktur, obiektów i funkcji. Interfejs ten stosowany jest w celu ułatwienia korzystania ze skomplikowanych funkcjonalności systemu, bibliotek i aplikacji poprzez mnóstwo skomplikowane, łatwe do użycia oraz jednoznaczne funkcje/metody.

W przypadku omawianej pracy będzie to zbiór funkcji wzajemnie powiązanych, komplementarnych w zakresie zadanej funkcjonalności o ścisłe określonych strukturach wyjściowych i wejściowych.

Biblioteka X2M została opracowana w celu zapewnienia łatwego i szybkiego korzystania z danych znajdujących się na platformie XNAT, nie ograniczając przy tym możliwości samej platformy. Platforma XNAT posiada zunifikowany schemat przechowywania danych, który jest taki sam na wszystkich instancjach znajdujących się w sieci. Dodatkowym argumentem przemawiającym za platformą XNAT jest powszechność i popularność w wielu projektach, badaniach oraz inicjatywach wymiany danych, między innymi CVRG (ang. Cardiovascular Research Grid), The Jackson Heart Study czy PREDICT-HD study of Huntington's Disease lista wszystkich pozostały w pozycji [31] bibliografii. Wybranym środowiskiem do implementacji rozwiązania jest środowisko Matlab, ponieważ jest powszechnie stosowane i popularne w wielu dziedzinach przetwarzania danych medycznych ze względu na swoją prostotę i gamę oferowanych funkcjonalności.

## 5.1 Opis funkcjonalności

Poszczególne funkcje biblioteki X2M zostały oparte na metodach stylu REST.[22]

W tabeli [5.1] przedstawione zostały przykłady metod REST z nazwami zasobów, na rzecz których zostały wykonane. Należy zaznaczyć, że w tabeli [5.1] początkowy URL do poszczególnych instancji systemu został pominięty celowo.

Tabela 5.1: Wykorzystane zasoby z podziałem na metody REST

Metoda	Zasób	Wykorzystanie w
<b>GET</b>	/data/users	X2MGetUsers
<b>GET</b>	/data/archive/scanners	X2MGetScanners
<b>GET</b>	/data/archive/projects	X2MGetProjects
<b>GET</b>	/data/archive/projects/{ID}/subjects	X2MGetSubjectsByProject
<b>GET</b>	/data/archive/projects/{ID}/subjects/{ID   Label}	X2MQuery
<b>GET</b>	/data/archive/subjects	X2MGetSubjects
<b>GET</b>	/data/projects/{ID}/subjects/{ID   Label}/experiments/{ID}/scans/{ID}/files	X2MDownloadData
<b>POST</b>	/data/search	X2MQuery

### 5.1.1 Przykład

Przykładowe wykorzystanie metody GET, odpowiadającej za pobranie listy wszystkich pacjentów z bazy platformy XNAT, jest następujące:

[GET] - <http://example.com/data/archive/subjects?format=json>

Gdzie :

- **http://example.com** – URL do danej instancji serwera/platformy XNAT,
- **?format=json** - format danych wyjściowych,
- **/data/archive/subjects** – zasób, na którym zapytanie zostało wykonane.

## 5.2 Zbiór funkcji

Poniżej przedstawione zostaną wszystkie funkcje wraz ze szczegółowym opisem, wchodzące w skład biblioteki X2M. W ramach każdej funkcji zostanie opisany zestaw danych wejściowych i wyjściowych, z dokładnym przedstawieniem struktur zawierających te dane. Funkcje te należy rozumieć jako pewne bloki projektowe, umożliwiające użytkownikowi zbudowanie własnego skryptu realizującego zadaną funkcjonalność. Funkcje biblioteki X2M można podzielić na trzy grupy ze względu na podprocesy, jakie realizują, które zostały opisane poniżej.

W ramach niniejszej pracy przyjęto konwencję notacji parametrów podczas wywoływania poszczególnych funkcji z rozbiciem na parametry wejściowe, wyjściowe i nazwy funkcji przedstawione poniżej:

- Nazwa funkcji - zawsze pogrubionym tekstem; w przypadku istnienia parametrów wyjściowych rozpoczyna się po znaku równości “=”, a kończy się lewym okrągłym nawiasem “(”.
- Parametry wejściowe - tekst pogrubiony struktury, podkreślony parametr opcjonalny, normalny parametr wejściowy jako wartość. Zaczynają się lewym okrągłym nawiasem “(” występującym po nazwie funkcji, a kończą się prawym okrągłym nawiasem “)”. Poszczególne parametry rozdzielane są od siebie przecinkiem.
- Parametry wyjściowe - tekst pisany kursywą struktury, podkreślony parametr opcjonalny, normalny parametr wyjściowy jako wartość. Parametry wyjściowe zaczynają się lewym kwadratowym nawiasem “[”, a kończą się prawym kwadratowym nawiasem “]”. Poszczególne parametry rozdzielane są od siebie przecinkiem. Należy zaznaczyć, że w przypadku, gdy zwracany jest tylko jeden parametr wyjściowy, nie ma potrzeby stosowania nawiasów kwadratowych.

### 5.2.1 Funkcje pomocnicze

Funkcjami pomocniczymi nazwane są funkcje, które same w sobie nie realizują procesu pobierania ani wyszukiwania danych. Są one jednak konieczne do poprawnego skonfigurowania środowiska oraz do ustalenia wymaganych parametrów takich jak na przykład lokalizacja sieciowa platformy XNAT, z której pobierane będą dane. Dodatkowo pozwalają na logowanie poszczególnych kroków i zapisywanie tych logów do lokalnych plików w formacie CSV.

#### 5.2.1.1 X2MSetPath

**X2MSetPath** jest funkcją służącą do ustalania ścieżki katalogu roboczego, to jest takiego, do którego pobierane będą wynikowe dane. Należy zaznaczyć również, że w tej lokalizacji powinien znajdować się plik *servers.mat* zawierający listę serwerów. Więcej o pliku w dodatku [A]. Postać funkcji jest następująca:

*[selpath,flagServers] = X2MSetPath* [5.1.1]

Gdzie:

- *selpath* - parametr wyjściowy typu string, zawierający lokalizację wskazanego katalogu. Katalog ten później będzie wykorzystywany jako miejsce zapisu danych i logów z działania programu,
- *flagServers* - parametr wyjściowy typu BOOL o wartości prawda (**1**), jeśli plik znajduje się we wskazanym katalogu i fałsz (**0**) w przeciwnym przypadku.

#### 5.2.1.2 X2MAAddServer

Funkcja **X2MAAddServer** umożliwia dodanie serwera (wpisu) do struktury pliku *servers.mat* więcej na temat pliku w dodatku [A]. Funkcja ta dodatkowo sprawdza, czy podane przez użytkownika dane są poprawne do zalogowania się do instancji platformy XNAT, poprzez wysłanie testowego zapytania. Postać funkcji jest następująca:

*servers = X2MAAddServer(selpath,servers)* [5.1.2]

Gdzie:

- *servers* - wyjściowa struktura zawierająca dane serwerów [A],
- **selpath** - parametr wejściowy typu string zawierający lokalizację katalogu, w którym należy stworzyć plik *servers.mat* albo zmodyfikować już istniejący; jest to zależne od tego czy plik istnieje w podanym katalogu,
- **servers** - opcjonalna struktura wejściowa, zawierająca dane dotyczące serwerów, podanie jej skutkuje dodaniem jej zawartości do pliku *servers.mat* w podanym katalogu.

Przykład działania funkcji **X2MAAddServer** został opisany w dodatku [B].

### 5.2.1.3 X2MAddToLog

**X2MAddToLog** jest funkcją, za pomocą której dodawany jest wpis do globalnej tabeli LOG. Tabela LOG może zostać później wykorzystana w funkcji **X2MPrintLog**, która powoduje zapisanie jej do lokalnego pliku w formacie CSV, który pozwala na późniejszą analizę sekwencji zdarzeń. Tabela LOG jest również dostępna do oglądu z dowolnego miejsca/kroku w skrypcie. Struktura tabeli LOG wraz z zawartością dla przypadku testowego została przedstawiona na rys. (5.1).

	A time	B action	C servers	D users	E errors	F querys	G subjects	H experiments	I numberOffiles
	Datetime...	▼TEXT	▼TEXT	▼NUMBER	▼TEXT	▼TEXT	▼TEXT	▼TEXT	▼TEXT
1	time	action	servers	users	errors	querys	subjects	experiments	numberOffiles
2	20:29:...	check	https://cz...	pszyma...	OK - Check conn...				
3	20:29:...	check	https://ce...	pszyma...	OK - Check conn...				
4	20:29:...	check	https://db...	pszyma...	OK - Check conn...				
5	20:30:...	query	https://cz...	pszyma...	send - query	YOB >= 1968 and YOB <= 1978			
6	20:30:...	query	https://cz...	pszyma...	query - OK	YOB >= 1968 and YOB <= 1978			0
7	20:30:...	query	https://ce...	pszyma...	send - query	YOB >= 1968 and YOB <= 1978			
8	20:30:...	query	https://ce...	pszyma...	query - OK	YOB >= 1968 and YOB <= 1978			2
9	20:30:...	query	https://ce...	pszyma...	query inner - ok	YOB >= 1968 and YOB <= 1978	CENTRAL_S00027		
10	20:30:...	query	https://ce...	pszyma...	query inner - ok	YOB >= 1968 and YOB <= 1978	CENTRAL_S00037		
11	20:30:...	query	https://db...	pszyma...	send - query	AGE >= 50 and AGE <= 40			
12	20:30:...	query	https://db...	pszyma...	query - OK	AGE >= 50 and AGE <= 40			7
13	20:30:...	query	https://db...	pszyma...	query inner - ok	AGE >= 50 and AGE <= 40	ConnectomeDB_...		
14	20:30:...	query	https://db...	pszyma...	query inner - ok	AGE >= 50 and AGE <= 40	ConnectomeDB_...		
15	20:31:...	query	https://db...	pszyma...	query inner - ok	AGE >= 50 and AGE <= 40	ConnectomeDB_...		

Rysunek 5.1: Struktura globalnej tabeli LOG. [źródło własne]

Postać funkcji **X2MAddToLog** jest następująca:

**X2MAddToLog(type,server,user,error,project,subject,experiment,query,numerOfFile)**

[5.1.3]

Gdzie:

- **type** - typ działania/akcji możliwe wartości to :
  - 'check' - sprawdzenie połączenia,
  - 'query' - zapytanie przeszukujące zasoby (pływkie wyszukiwanie),
  - 'inner query' - podzapytanie przeszukujące wybrane pierwotne zasoby wybrane przez 'query' (głębokie wyszukiwanie),
  - 'download' - pobieranie danych.
- **server** - instancja serwera, na rzecz którego akcja została przeprowadzona,
- **user** - użytkownik, który przeprowadzał akcję,
- **error** - ewentualny błąd wraz z opisem lub, w przypadku sukcesu, nazwa akcji i sufiks 'OK' na przykład:
  - 'query OK' - pomyślnie wykonane zapytanie,

- 'error The server returned the message: Not Found for URL HTTP 404' - błąd zapytania, zasób nie istnieje; więcej o kodach błędu protokołu HTTP w pozycji [27] bibliografii.

- **query** - zadane zapytanie, przykład widoczny na rys. (5.1),
- **subject** - unikalny identyfikator (ID) pacjenta,
- **experiment** - unikalne identyfikator (ID) badania,
- **numerOfFile** - liczba pobranych/wyszukanych plików/zasobów.

#### 5.2.1.4 X2MPrintLog

**X2MPrintLog** to funkcja służąca do zapisania globalnej tabeli do lokalnego pliku CSV; zawartość pliku jest co do struktury taka sama, jak w przypadku funkcji **X2MAddToLog**. Plik ten zapisywany jest pod nazwą *log\_RRRR\_MM\_DD\_GG\_MM.CSV*, w podanej w parametrze wejściowym ścieżce.

Postać funkcji jest następująca:

**X2MPrintLog(selpath)**

[5.1.4]

Gdzie:

- **selpath** - to ścieżka do katalogu, w którym zostanie zapisany plik CSV, jest to parametr obowiązkowy.

Zalecane jest wywoływanie funkcji **X2MPrintLog** zawsze po pełnym przebiegu skryptu, w celu zapisana wywołań poszczególnych sekwencji w lokalnym pliku. Budowa tabeli Log, a co za tym idzie wynikowego pliku CSV, pozwala na szybkie ustalanie miejsca i przyczyny ewentualnych błędów w poszczególnych krokach. Dodatkowo zapisana w ten sposób sekwencja jest prosta do odtworzenia.

#### 5.2.1.5 X2MSendMail

Funkcja **X2MSendMail** odpowiedzialna jest za wysyłanie maila o zakończeniu procesu. Zastosowana jest w przypadku procesu pobierania i wyszukiwania przedstawionego w sekcji **Query - Auto** (6.4.2). Można też wywołać ją na końcu danego skryptu w celu powiadomienia użytkownika o zakończeniu wywołania.

Postać funkcji jest następująca:

**X2MSendMail(recipient\_email)**

[5.1.5]

Gdzie:

- **recipient\_email** - adres email, na który dany mail ma zostać wysłany.

Nadawcą maila jest *xnat.spm.connector@gmail.com*, który wysyła maila o treści:

*X2M script job - Hello! Your job is done! Hope for best*

### 5.2.1.6 X2MCheckConnection

**x2mCheckConnection** jest funkcją służącą do sprawdzenia, czy podane przez użytkownika dane do logowania są poprawne na instancji platformy XNAT. Funkcja ta bazuje na wysłaniu zapytania autoryzacyjnego do platformy i sprawdza kod odpowiedzi HTTP. W przypadku, gdy prefiks kod odpowiedzi jest różny od '20' zgłoszany jest błąd. Typowe kody błędu:

- 401 - użytkownik nie posiada konta na podanej instancji platformy,
- 403 - użytkownik nie ma uprawnień, aby wykonywać funkcje RESTowe na danych zasobach,
- 400 lub 404 - niepoprawny Host lub zasób.

Postać funkcji **x2mCheckConnection** jest następująca:

**x2mCheckConnection(url,user,password)**

[5.1.6]

Gdzie:

- url - URL do instancji platformy XNAT,
- user - użytkownik znajdujący się na zadanej instancji platformy XNAT,
- password - hasło użytkownika do logowania się na zadanej platformie XNAT.

### 5.2.2 Funkcje wyszukujące

Funkcje wyszukujące bazują na metodzie GET protokołu HTTP, wykorzystywane są do odpytywania platformy XNAT o określone zasoby. Zasoby te później mogą zostać wykorzystane w procesie pobierania danych, realizowanym przez **Funkcje pobierania** opisane w sekcji (5.2.3).

#### 5.2.2.1 X2MGetUsers

Funkcja **X2MGetUsers** zwraca listę wszystkich użytkowników znajdujących się na zadanym serwerze, na rzecz którego wykonane zostało zapytanie.

Postać funkcji jest następująca:

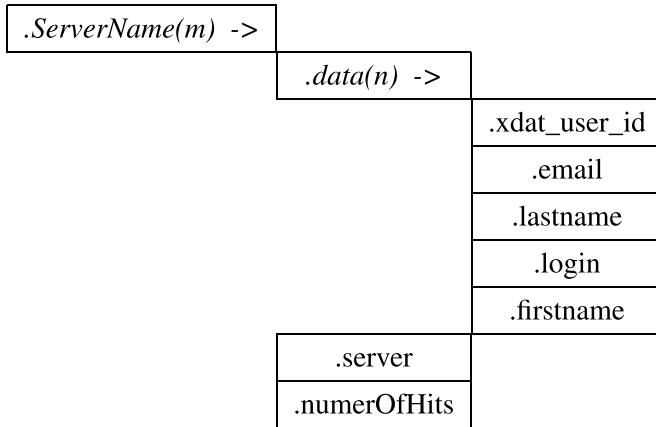
**users = X2MGetUsers(servers)**

[5.2.1]

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **users** - wynikowa struktura zawierająca dane użytkowników, o postaci przedstawionej w tabeli [5.2].

Tabela 5.2: Zawartość struktury wyjściowej *users*



Elementy w tabeli [5.2] oznaczają:

- (m) - liczba serwerów w strukturze wejściowej w relacji 1:\*,
- (n) - liczba użytkowników w strukturze w relacji 1:\*,
- *.ServersName(m)* - pseudo nazwa serwera uzyskana z URL do podanej w parametrze wejściowej platformy XNAT, na przykład dla <https://czarnobyl.ibib.waw.pl> będzie to Czarnobyl,
- *.data(n)* - struktura danych dla danego użytkownika zawierająca pola przedstawione w tabeli [5.3],
- *.server* - URL do serwera, na którym wykonana została funkcja,
- *.numerOfHits* - liczba unikalnych użytkowników (dla zadanego serwera) .

Tabela 5.3: Zawartość struktury *data(n)*, dla danego użytkownika

<b>xdat_user_id</b>	unikalny numer identyfikator (ID) użytkownika
<b>login</b>	login
<b>email</b>	adres email
<b>firstname</b>	imie
<b>lastname</b>	nazwisko

### 5.2.2.2 X2MGetScanners

**X2MGetScanners** jest funkcją, która zwraca listę wszystkich skanerów. Skanerami nazywane są wszystkie urządzenia, z których pochodzą obrazowe dane medyczne w ramach danych badań znajdujących się na zadany serwerze.

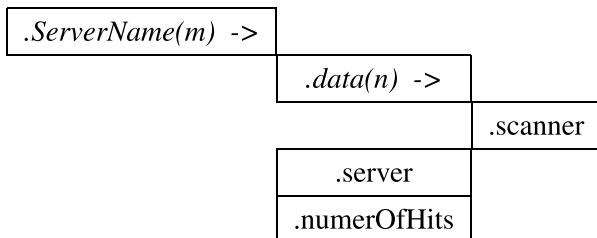
Postać funkcji jest następująca:

$$scanners = \mathbf{X2MGetScanners}(servers) \quad [5.2.2]$$

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **scanners** - wynikowa struktura zawierająca dane użytkowników, o strukturze przedstawionej w tabeli [5.4].

Tabela 5.4: Zawartość wyjściowej struktury *scanners*



Gdzie:

- (m) - liczba serwerów w strukturze wejściowej w relacji 1:\*,
- (n) - liczba skanerów w strukturze w relacji 1:\*,
- *.ServersName(m)* - pseudo nazwa serwera uzyskana z URL do podanej w parametrze wejściowej platformy XNAT, na przykład dla <https://czarnobyl.ibib.waw.pl> będzie to Czarnobyl,
- *.scanner* - nazwa skanera wewnętrz struktury *.data(m)* na zadany serwerze,
- *.server* - URL do serwera, na którym wykonana została funkcja,
- *.numberOfHits* - liczba unikalnych skanerów (dla zadanego serwera).

### 5.2.2.3 X2MGetProjects

Funkcja **X2MGetProjects** zwraca listę wszystkich projektów znajdujących się na zadany serwerze. Projekty rozumiane są jako nadrzędne elementy w bazie danych, a badania są elementem tworząnym w ramach projektu.

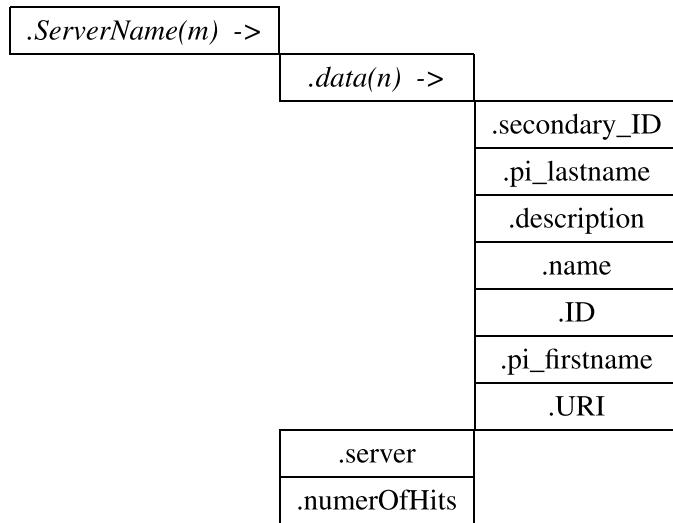
Postać funkcji jest następująca:

$$projects = \mathbf{X2MGetProjects}(servers) \quad [5.2.3]$$

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **projects** - wynikowa struktura zawierająca dane projektów, o postaci przedstawionej w tabeli [5.5].

Tabela 5.5: Zawartość wyjściowej struktury *projects*



Gdzie:

- (m) - liczba serwerów w strukturze wejściowej w relacji 1: $*$ ,
- (n) - liczba skanerów w strukturze w relacji 1: $*$ ,
- *.ServerName(m)* - pseudo nazwa serwera uzyskana z URL do podanej w parametrze wejściowej platformy XNAT, na przykład dla <https://czarnobyl.ibib.waw.pl> będzie to Czarnobyl,
- *.data(n)* - struktura danych dla danego projektu, o strukturze przedstawionej w tabeli [5.6],
- *.server* - URL do serwera, na którym wykonana została funkcja,
- *.numberOfHits* - liczba unikalnych skanerów (dla zadanego serwera).

Tabela 5.6: Zawartość struktury *.data(n)*, dla danego projektu

<b>ID</b>	unikalny identyfikator (ID) projektu
<b>secondary_ID</b>	dodatkowy identyfikator (ID) projektu
<b>pi_firstname</b>	imię twórcy projektu
<b>pi_lastname</b>	nazwisko twórcy projektu
<b>description</b>	opis projektu
<b>name</b>	nazwa projektu
<b>URI</b>	URI do zasobu

#### 5.2.2.4 X2MGetSubjects

Funkcja **X2MGetSubjects** zwraca listę wszystkich pacjentów znajdujących się na zadanym serwerze. Zwracana lista, poza samymi identyfikatorami pacjentów, zawiera ich podstawowe dane przedstawione w tabeli [5.7] i [5.8]. Za pomocą funkcji **X2MGetSubjects** dodatkowo możliwe jest pobranie szczegółowych danych dla określonej przez użytkownika liczby pacjentów. Ograniczenie liczby pacjentów jest istotne w aspekcie czasu, który jest potrzebny do wykonania funkcji.

Postać funkcji jest następująca:

*[ dataSubjects, dataSubjectsDetailed ] = X2MGetSubjects(servers, upTo)* [5.2.4]

Gdzie:

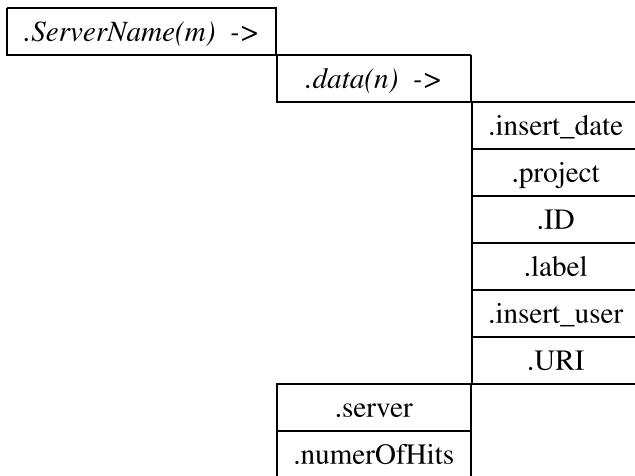
- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- *dataSubjects* - struktura wyjściowa zawierająca podstawowe dane pacjentów przedstawiona w tabeli [5.7],
- *dataSubjectsDetailed* - opcjonalna struktura wyjściowa zawierająca szczegółowe dane pacjentów, zależna od parametru upTo, o budowie przedstawionej w tabeli [5.9],
- upTo – opcjonalny parametr liczbowy określający maksymalną liczbę pacjentów, dla których zwrócone mają zostać szczegółowe dane, określana jest dla całkowitej liczby pacjentów. W przypadku, gdy parametr ten jest mniejszy niż sumaryczna liczba pacjentów na serwerze, szczegółowe dane będą pobierane w kolejności, w jakiej serwery zostały dodane.

Należy rozróżnić trzy wartości parametru upTo:

- *pusta lub niewypełnione* – struktura *dataSubjectsDetailed* będzie pusta, czyli dla żadnego pacjenta nie zostały pobrane szczegółowe dane,
- *X* – gdzie X to liczba, oznacza, że maksymalnie dla X pacjentów pobrane zostaną szczegółowe dane. Jeśli podana wartość będzie większa niż liczba pacjentów znajdująca się łącznie na wszystkich serwerach, to zostanie ograniczona do sumacyjnej liczby pacjentów,
- *0* – dla każdego pacjenta mają zostać pobrane szczegółowe dane.

## *dataSubjects*

Tabela 5.7: Zawartość wyjściowej struktury *dataSubjects*



Gdzie:

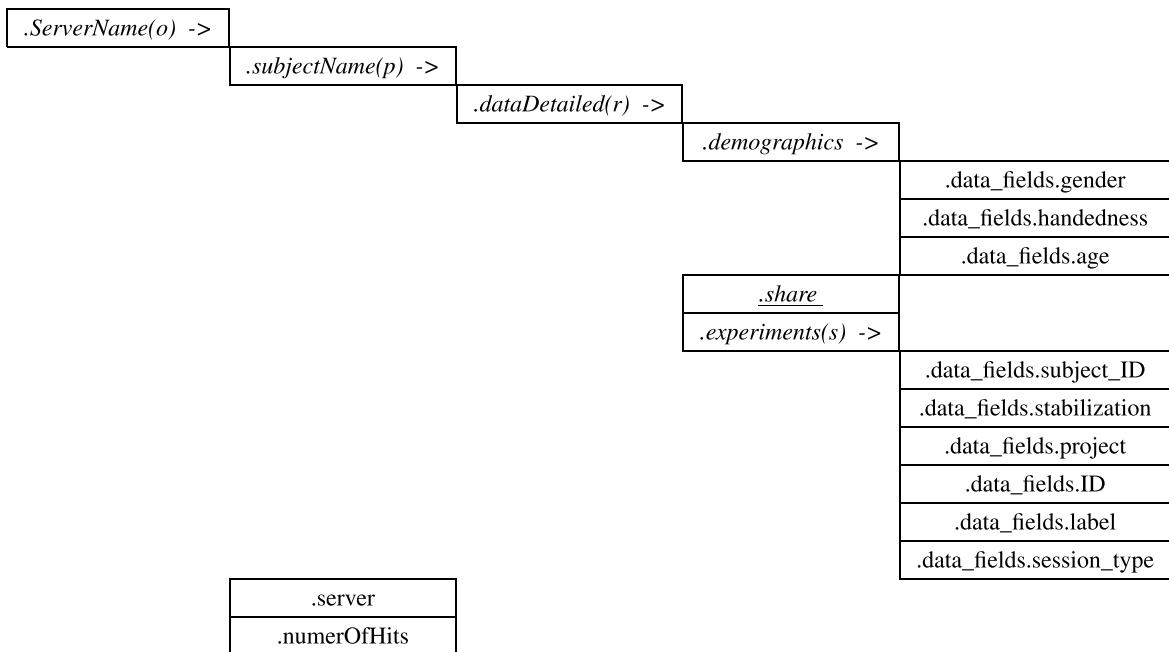
- *.ServerName(m)* to nazwa serwera uzyskana z URL do danego serwera, na przykład dla <https://czarnobyl.ibib.waw.pl> będzie to Czarnobyl,
- (m) – to liczba serwerów w strukturze ; świadczy o relacji 1:\*,
- *.data(n)* to struktura danych dla danego przedstawiona w tabeli [5.8],
- (n) – to liczba pacjentów w strukturze ; świadczy o relacji 1:\*,
- *.server* to URL do serwera, na którym wykonana została funkcja,
- *.numberOfHits* to liczba unikalnych pacjentów (dla zadanego serwera).

Tabela 5.8: Zawartość struktury *.data(n)*, dla danego pacjenta

<b>insert_date</b>	data dodania pacjenta w formacie RRRR-MM-DD GG:MM:SS.0
<b>project</b>	identyfikator (ID) projektu do którego został przypisany pacjent
<b>ID</b>	unikalny identyfikator (ID) pacjenta
<b>label</b>	unikalny znacznik pacjenta
<b>insert_user</b>	login użytkownika wprowadzającego
<b>URI</b>	URI do zasobu

## *dataSubjectsDetailed*

Tabela 5.9: Zawartość opcjonalnej wyjściowej struktury *dataSubjectsDetailed*



Gdzie:

- *.ServerName(o)* - nazwa serwera uzyskana z URL do danego serwera, na przykład dla <https://czarnobyl.ibib.waw.pl> będzie to Czarnobyl,
- (o) – liczba serwerów w strukturze ; świadczy o relacji 1:\*,
- *.subjectName(p)* - unikalny identyfikator (ID) pacjenta,
- (p) – liczba pacjentów, dla których zostały pobrane szczegółowe dane,
- *.dataDetailed(r)* - struktura danych szczegółowych rozdzielająca się na trzy podstruktury:
  - *.demographics*,
  - *.share*(optional),
  - *.more experiment(s)*.
- (r) – liczba podstruktur dla danego pacjenta, maksymalnie 3. Struktura *.sharing/share* jest opcjonalna i nie zawsze podawana dla danego pacjenta,
- *.demographics* – podstruktura danych demograficznych pacjenta o postaci przedstawionej w tabeli [5.10],
- *.share* – opcjonalna struktura zawierająca metadane związane z przechowywaniem danych na serwerze,
- *.experiment(s)* - podstruktura danych dotyczących badań pacjenta o postaci przedstawionej w tabeli [5.11],
- (s) – liczba badań dla danego pacjenta ; świadczy o relacji 1:\*,
- *.server* - URL do serwera, na którym wykonana została funkcja,
- *.numerOfHits* - liczba unikalnych pacjentów (dla zadanego serwera) .

Tabela 5.10: Zawartość podstruktury *.demographics*, dane demograficzne

<b>data_fields.gender</b>	płeć
<b>data_fields.handedness</b>	lateralizacja/ręczność
<b>data_fields.age</b>	wiek

Tabela 5.11: Zawartość podstruktury *.experiment(s)*, dane badań danego pacjenta

<b>.data_fields.subject_ID</b>	unikalny identyfikator (ID) pacjenta
<b>.data_fields.stabilization</b>	stabilizacja, która została wykorzystana podczas badania np. Head
<b>.data_fields.project</b>	identyfikator (ID) projektu, w ramach którego zostało przeprowadzone badanie
<b>.data_fields.ID</b>	unikalne identyfikator (ID) badania w ramach projektu
<b>.data_fields.label</b>	unikalny znacznik badania
<b>.data_fields.session_type</b>	typ sesji podczas badania

#### 5.2.2.5 X2MGetSubjectsByProject

Funkcja **X2MGetSubjectsByProject** zwraca listę wszystkich identyfikatorów dotyczących pacjentów znajdujących się na zadanym serwerze, pogrupowanych względem projektu. Postać funkcji jest następująca:

[subjectsByProjectData,projectsData] = **X2MGetSubjectsByProject(servers,projectsData)** [5.2.5]

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **subjectsByProjectData** - to struktura wyjściowa zawierająca identyfikatory pacjentów pogrupowanych względem projektów,
- **projectsData** - to struktura zawierająca dane projektu, taka sama co do budowy jak struktura wyjściowa z funkcji **X2MgetProjects**. Należy zauważyć, że jest jednocześnie opcjonalnym parametrem wejściowym, jak i opcjonalnym parametrem wyjściowym. W przypadku niepodania tej struktury jako parametru wejściowego program sam pobiera listę wszystkich projektów na zadanym serwerze i pobiera dane dotyczące wszystkich pacjentów dla danego projektu (wynikiem będzie struktura zawierająca wszystkie projekty z wszystkimi pacjentami). Natomiast gdy struktura ta jest podana jako parametr wejściowy, pobrane zostaną dane odnoszące się tylko do pacjentów będących w przekazanych projektach.

### 5.2.2.6 X2MGetSubjectsFromProject

**X2MGetSubjectsFromProject** jest funkcją służącą do pobrania listy identyfikatorów wszystkich pacjentów wraz z danymi odnoszącymi się do nich, w ramach projektu podanego w parametrach wejściowych. Należy zaznaczyć, że nie będą to dane obrazowe, a dane opisujące pacjenta, takie jak na przykład wiek, płeć czy unikalny identyfikator (ID) pacjenta. Pełna struktura danych widoczna w tabeli [5.7] i [5.8]. Postać funkcji jest następująca:

*[dataSubjects,dataSubjectsDetailed] = X2MGetSubjectsFromProject(servers,project,upTo)* [5.2.6]

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **project** - nazwa pożdanego projektu zapisana jako string; jest to parametry wymagany.

Pozostałe parametry wejściowe i struktury spójne w działaniu i budowie jak te z **X2MGetSubjects**.

### 5.2.2.7 X2MQuery

Funkcja **X2MQuery** wykorzystywana jest w celu wyszukania pacjentów. Parametrami możliwymi do wyszukiwania są wiek (dokładny), wiek (jako przedział) i płeć. Istotą działania funkcji jest odpytanie bazy danych XNAT o szukane parametry, proces ten nazywany później jest pierwotnym Query. W tym celu wykorzystywana jest metoda POST, za pośrednictwem której wysyłany jest plik XML. Wymieniony plik zawiera wyszukiwane wartości i listę pól, które mają zostać zwrocone; przykładowy plik został podany w dodatku [C]. W wyniku pierwotnego Querry zwracane są identyfikatory (ID) pacjentów spełniających kryteria wyszukiwania. Niestety sam identyfikator (ID) pacjenta nie wystarczy do pobrania jego danych, wynika to z budowy encji platformy XNAT. Należy więc ponownie odpytać bazę o bardziej szczegółowe dane pacjenta, takie jak projekty czy badania, w których pacjent brał udział. Dopiero te dane pozwalają na jednoznaczne ustalenie zasobu, który należy odpytać w celu pobrania danych.

Postać funkcji jest następująca:

*[data,found,modality,servers,projects] = X2MQuery(servers,age\_from,age\_to,sex,LoIn)* [5.2.7]

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **age\_from** - opcjonalny parametr wejściowy typu numeric, zależny od **LoIn**, oznaczający wiek od lub dokładny wiek,
- **age\_to** - opcjonalny parametr wejściowy typu numeric, wykorzystywany tylko, gdy **LoIn** = 1 i oznaczający wiek do,

- sex - opcjonalny parametr typu numeric, oznaczający płeć o wartościach:
  - **1** lub nie podanie - dowolna,
  - **2** - nieznana,
  - **3** - męska,
  - **4** - żeńska,
  - **5** - inna (np. fantom).
- loIn - opcjonalny parametr typu numeric, sterujący przeszukiwanym wiekiem w przypadku, gdy jego wartość wynosi **1** - przeszukiwany jest zakres od age\_from do age\_to, jeśli jego wartość jest różna od **1** wtedy parametr wyszukiwany wiek równy jest parametrowi age\_from.
- *data* - struktura wyjściowa zawierająca dane pacjentów, których dane spełniły warunki wyszukiwania. Struktura ta jest spójna ze strukturą *dataSubjectsDetailed* przedstawioną w tabeli [5.9],
- *found* - parametr wyjściowy typu numeric oznaczający liczbę unikalnych pacjentów, których dane spełniły warunki wyszukiwania,
- *modality* - lista wyjściowa zawierająca wszystkie unikalne modalności, które zostały wyszukane w danych pacjentów spełniających warunki wyszukiwania,
- *servers* - struktura wyjściowa zawierający dane serwerów z informacją, ilu pacjentów zostało wyszukanych dla danego serwera,
- *projects* - struktura wyjściowa zawierająca wszystkie unikalne projekty, które zostały wyszukane w danych pacjentów spełniających warunki wyszukiwania.

### 5.2.3 Funkcje pobierania

Funkcje pobierania bazują na metodzie GET protokołu HTTP i odpowiedzialne są za pobieranie danych dotyczących pacjentów, takich jak na przykład obrazy. Zasób, na rzecz którego metoda ma być wykonana, jest dynamicznie ustalany na podstawie danych z funkcji wyszukujących, opisanych w sekcji **Funkcje wyszukujące** (5.2.2).

#### 5.2.3.1 X2MDownloadDataSubjectNoGui

**X2MDownloadDataSubjectNoGui** jest funkcją, która pobiera dane dla zadanego pacjenta w ramach danego badania i projektu. Należy zaznaczyć, że dane są tu rozumiane jako wszystkie dane dotyczące danego pacjenta, takie jak obrazy w formacie DICOM, NIfTI oraz dodatkowe dane w dowolnym formacie, które odnoszą się do tego pacjenta. Dane pobierane są do lokalizacji roboczej podowanej przez użytkownika podczas działania programu, gdzie później są rozpakowywane oraz zapisywany jest log w postaci pliku CSV, który zawiera listę poszczególnych operacji wykonanych podczas działania programu.

**X2MDownloadDataSubjectNoGui(servers,dataSubjectsDetailed)** [5.3.1]

Gdzie:

- **servers** - wejściowa struktura zawierająca dane serwerów [A],
- **dataSubjectsDetailed** - struktura wejściowa zawierająca szczegółowe dane pacjentów. Jest to struktura wyjściowa z funkcji **X2MGetSubjectsFromProject** i **X2MGetSubjects**.

#### 5.2.3.2 X2MDownloadData

**X2MDownloadData** to funkcja, która odpowiada za pobieranie danych dla zadanego w parametrach wejściowych pacjentów. Tak samo jak w przypadku funkcji opisanej w podsekcji **X2MDownloadDataSubjectNoGui** (5.2.3.1), pobierane dane są tu rozumiane jako wszystkie dane dotyczące danego pacjenta, takie jak obrazy w formacie DICOM, NIfTI oraz dodatkowe dane w dowolnym formacie, które odnoszą się do tego pacjenta. Podobnie jak w przypadku funkcji **X2MDownloadDataSubjectNoGui**, dane pobierane są do lokalizacji roboczej podowanej przez użytkownika podczas działania programu, gdzie później są rozpakowywane oraz zapisywany jest log w postaci pliku CSV, który zawiera listę poszczególnych operacji wykonanych podczas działania programu. W przeciwieństwie do funkcji **X2MDownloadDataSubjectNoGui**, funkcja **X2MDownloadData** pozwala na większą dowolność wyboru danych, które mają zostać pobrane. Lista tych parametrów wraz z opisem zostanie przedstawiona poniżej.

success = **X2MDownloadData(data,maxSubjects,regex,serversConnected,type,projects)** [5.3.2]

Gdzie:

- success - wyjściowy parametr typu BOOL, określający czy dane dotyczące podanego pacjenta zostały pobrane; o wartości prawda ("X") lub fałsz ( ' ),

- **data** - wejściowa struktura danych zawierająca między innymi unikalny identyfikator pacjenta, numer/nazwę badania i identyfikator projektu, w ramach którego dane badanie zostało wykonane, przedstawiona jest w tabeli [5.9]. Służy do zidentyfikowania zasobu na serwerze, z którego należy pobrać dane. Jest strukturą wyjściową z funkcji **X2MGetSubjects**, **X2MGetSubjectsByProject**, **X2MGetSubjectsFromProject**.
- maxSubjects - opcjonalny parametr wejściowy typu liczba całkowita (ang. integer), rozumiany jako maksymalna liczba pobranych unikalnych danych dotyczących pacjentów.
- regex - opcjonalny parametr wejściowy typu string definiujący szukany ciąg znaków w projekcie/badaniu na przykład 'fMRI'. W przypadku, gdy jest niewypełniony lub zawiera pusty ciąg, pozwala na pobranie danych z dowolnym opisem. W przeciwnym wypadku jego wartość wyszukiwana jest w opisach badań, pobierane są tylko badania zawierające podany ciąg znaków.
- type - opcjonalny parametr wejściowy definiujący tryb pracy przyjmuje wartości "1" i "0" jako liczby całkowite. Wykorzystywany jest jedynie podczas działania w GUI, gdzie dla wartości "1" nazywany jest trybem manualnym i powoduje wyświetlanie logu w oknie po lecęń środowiska Matlab, w przypadku "0" log nie jest wypisywany.
- projects - opcjonalna struktura wejściowa definiująca nazwy/identyfikatory projektów, z których dane mają być pobierane. Podanie nazw projektów w tej strukturze pozwala na pobranie danych jedynie z porządkowych projektów. Należy zaznaczyć, że nie podanie jej jako parametru wejściowego lub podanie go jako pustej struktury spowoduje pobranie wszystkich danych podanych w strukturze **data**, niezależne od tego, z jakiego projektu pochodzą.
- **serversConnected** - wejściowa struktura zawierająca dane serwerów, które są aktywne w czasie działania procesu pobierania o postaci przedstawionej w tabeli [5.12].

Tabela 5.12: Zawartość struktury **serversConnected**, połączone instancje platformy XNAT

<b>.name</b>	URL do instancji platformy XNAT
<b>.user</b>	Nazwa użytkownika
<b>.password</b>	Hasło
<b>.Connect</b>	Czy platforma jest dostępna tak - wartość 'X' nie - wartość ''

## 6 Aplikacja użytkownika - GUI

Poniżej przedstawione zostaną możliwości wraz ze szczegółowym opisem, przykładowej aplikacji w postaci GUI (ang. Graphic User Interface) zbudowanej na podstawie funkcji z biblioteki X2M. Aplikacja ta została opracowana w celu zademonstrowania możliwości biblioteki i realizuje ona funkcję łączenia się z wielomainstancjami platformy XNAT, przeszukiwania zasobów znajdujących się na nich i ostatecznie pobierania danych dotyczących określonych pacjentów. Możliwymi do przeszukiwaniami parametrami są:

- wiek (dokładny),
- wiek (zakres),
- płeć,
- metadane (dotyczące badania opisane później),
- nazwa projektu.

Należy zaznaczyć, że ilość pobieranych danych może zostać ograniczona do pewnej liczby pacjentów, a aplikacja umożliwia użytkownikowi pracę w dwóch trybach - manualnym i automatycznym. Aby uruchomić GUI, konieczne jest posiadanie programu Matlab w wersji co najmniej 2014a oraz pobranie biblioteki X2M z repozytorium GitHub, znajdującej się pod linkiem:

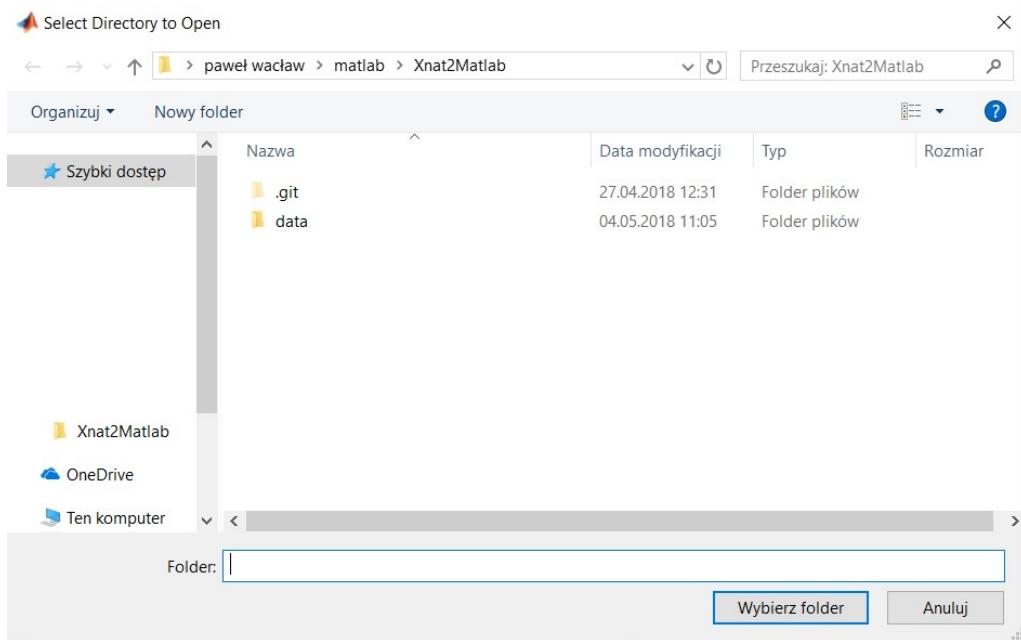
<https://github.com/Pszyman2/X2M>

Pobraną bibliotekę należy zapisać i rozpakować w dogodnej lokalizacji i dodać ją do biblioteki plików/funkcji wykonywalnych środowiska Matlaba, można to zrealizować za pomocą standardowej dla Matlaba funkcji **addpath**. Niżej opisane podrozdziały należy traktować jako instrukcję obsługi GUI. Na końcu każdego z podrozdziałów wymieniono wykorzystane w ramach niego funkcje biblioteki X2M. Funkcja **X2MAAddToLog** odpowiedzialna za zapis sekwencji wykonywanych działań wykorzystywana jest w większości funkcji biblioteki X2M, dlatego została ona celowo pominięta w poniższych podrozdziałach.

### 6.1 Uruchomienie - GUI

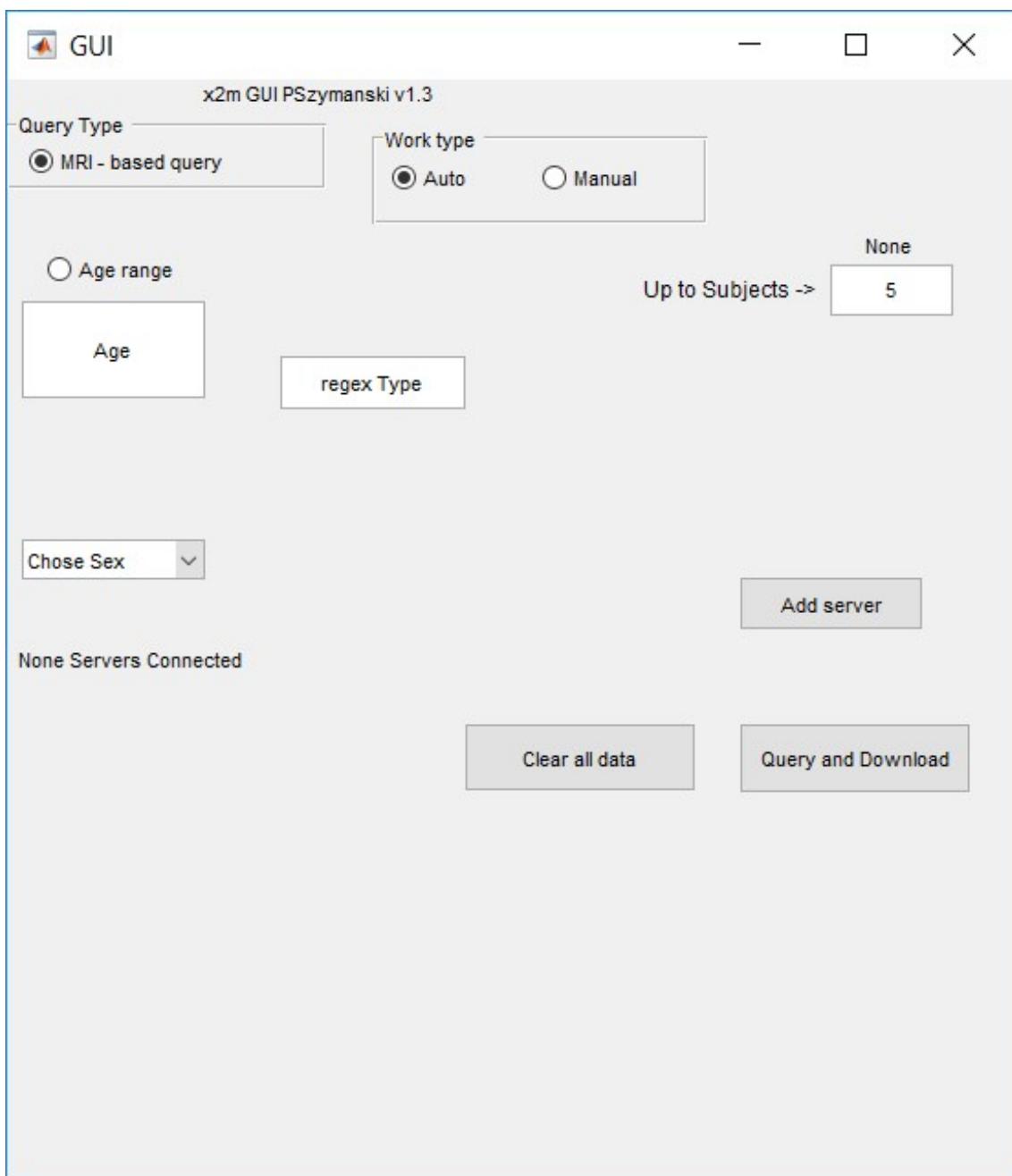
Uruchomienie GUI może zostać przeprowadzone na dwa sposoby - za pomocą dwukrotnego kliknięcia na plik *x2mGui.m* i uruchomienie go lub poprzez wpisanie *x2mGui* w linii poleceń środowiska Matlab. Podczas uruchomienia użytkownik zostanie poproszony o podanie lokalizacji do ścieżki roboczej rys. (6.1), w której zapisywane będą pobrane dane, logi i plik konfiguracyjny *servers.mat*. Wywoływane jest to za pośrednictwem funkcji **X2MSetPath**. Plik *servers.mat* zawiera dane dotyczące instancji platformy XNAT takie jak login i hasło oraz URL do platformy.

Jeśli w danej lokalizacji istnieje już plik *servers.mat*, wczytane zostaną z niego dane dotyczące instancji platformy XNAT i istnieje możliwość pominięcia punktu z podrozdziału **Dodawanie serwerów - GUI** (6.2). Wczytywane dane z pliku *servers.mat* są weryfikowane za pomocą funkcji **X2MCheckConnection**, a odpowiednia odpowiedź o pomyślnym połączeniu z platformą lub niepowodzeniu zapisywana jest do logu funkcją **X2MAddToLog**.



Rysunek 6.1: Ustalanie ścieżki roboczej. [źródło własne]

Po ustaleniu wyżej opisanej ścieżki wyświetlane zostanie właściwe GUI rys. (6.2).



Rysunek 6.2: Główny ekran GUI. [źródło własne]

Wykorzystywane funkcje :

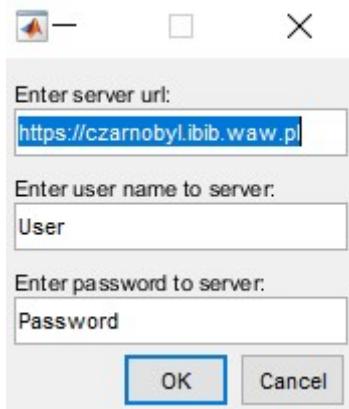
**X2MSetPath,**

**X2MCheckConnection.**

## 6.2 Dodawanie serwerów - GUI

Jest to krok opcjonalny, jeśli użytkownik posiada wypełniony plik servers.mat.

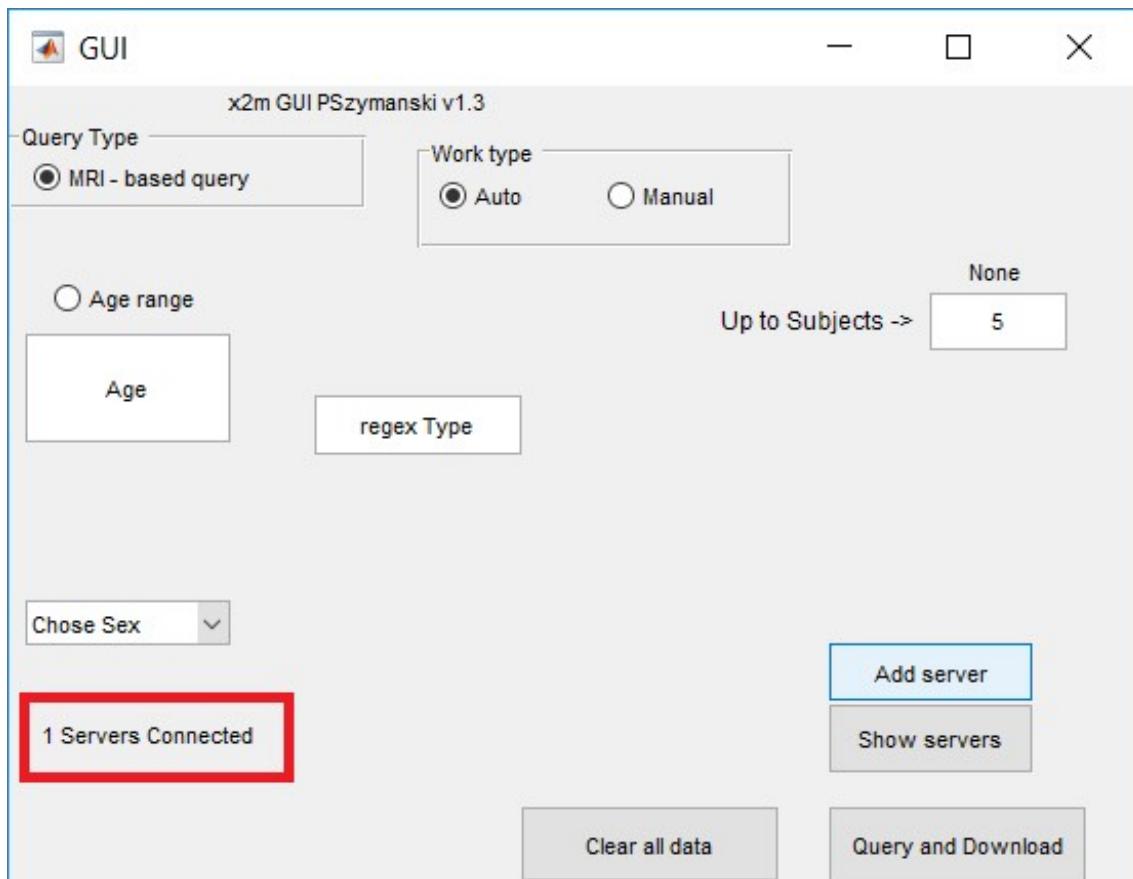
W celu poprawnego działania funkcjonalności biblioteki należy skonfigurować środowisko, a w szczególności podać dane do logowania się na danych instancjach platformy XNAT, nazywanej później serwerami. Istnieje możliwość wykonania tego na dwa sposoby - jeden opisany w dodatku [B], drugi natomiast z poziomu GUI. Aby dodać serwery z poziomu GUI należy kliknąć przycisk **Add server**. Następnie użytkownik musi wprowadzić dane do logowania na danym serwerze takie jak URL do serwera, User (login) i Password (hasło) rys. (6.3).



Rysunek 6.3: Dodawanie serwerów z poziomu GUI. [źródło własne]

Funkcja biblioteki **X2MAddServer** odpowiedzialna za dodawanie serwerów do pliku konfiguracyjnego *servers.mat* sprawdza, czy użytkownik podał poprawne dane poprzez wysłanie zapytania autoryzacyjnego realizowanego przez funkcje **X2MCheckConnection**. Jeżeli dane nie były poprawne, użytkownik zostanie poinformowany wiadomością, na przykład *HTTP 401 - XNATMat-Tool:HttpErrorThe server returned the message: "Unauthorized" for URL*, a podane przez użytkownika dane nie zostaną zapisane do pliku konfiguracyjnego.Więcej o kodach odpowiedzi w podsekcji **X2MCheckConnection** (5.2.1.6) i pozycji [27] bibliografii. W przypadku, gdy dane były poprawne, serwer zostanie dodany do pliku *servers.mat* i na GUI pojawi się informacja o połączenie do serwera zaznaczona czerwonym kolorem na rys. (6.4). Niezależnie od tego, czy platforma została dodana, czy nie za pomocą funkcji **X2MAddToLog** do logu zostanie dodany odpowiedni wpis zawierający ewentualne błędy.

Aby dodać kolejny serwer, należy postąpić analogicznie jak w przypadku pierwszego serwera. Pokazany na rys. (6.4) licznik będzie się zwiększał i będzie pokazywał liczbę połączonych serwerów. Należy zaznaczyć, że liczba ta może być mniejsza niż liczba serwerów znajdujących się w pliku *servers.mat*, ponieważ dany serwer może być aktualnie niedostępny, podany użytkownik może stracić uprawnienia do logowania się na nim lub hasło użytkownika może się zmienić/wygasnąć.



Rysunek 6.4: Pomyślne dodanie serwera. [źródło własne]

Dodatkowo należy zaznaczyć, że jeśli w danej lokalizacji zostanie skonfigurowany plik *servers.mat* lub zostanie przeniesiony z innej lokalizacji, krok ten można pominąć, ponieważ dane serwerów zostaną wczytane ze znajdującego się w tej lokalizacji pliku *servers.mat*.

Wykorzystywane funkcje biblioteki:

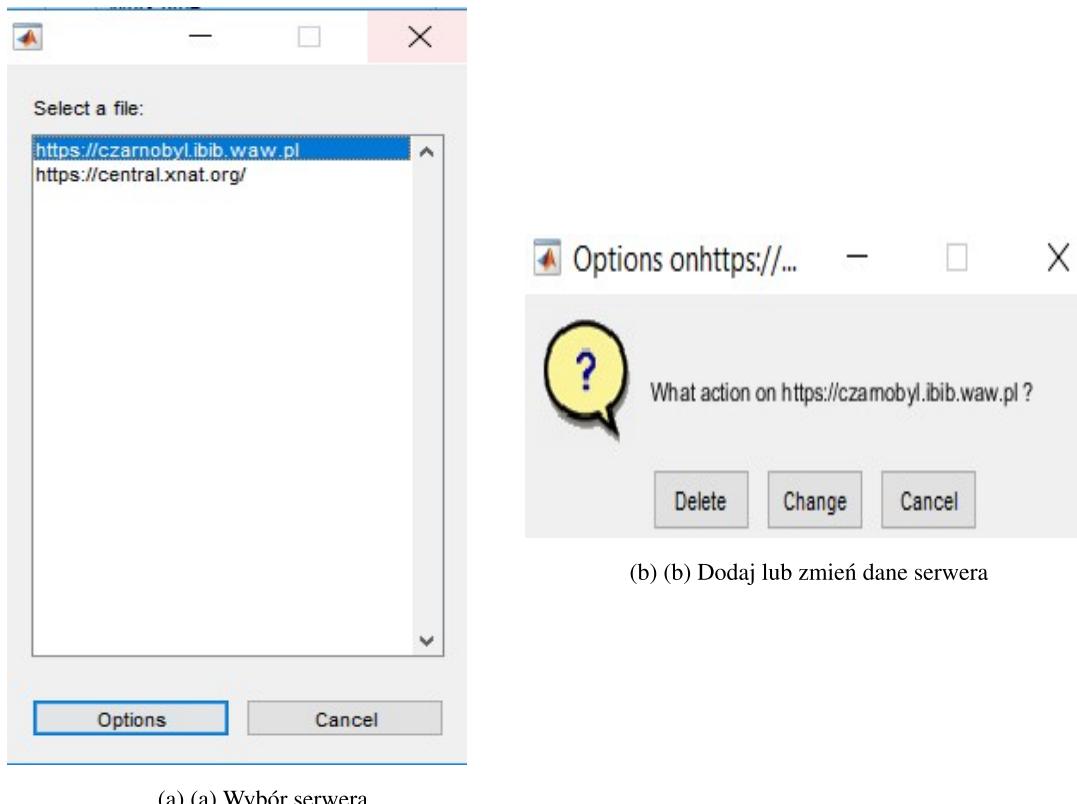
**X2MAddServer,**

**X2MCheckConnection.**

### 6.3 Modyfikacja serwerów - GUI

*Jest to krok opcjonalny.*

Jeśli istnieje potrzeba, aby dany serwer usunąć lub zmodyfikować, to jest to możliwe poprzez wcisnięcie przycisku **Show servers**, zaznaczenie porządkiego serwera rys. (6.5a), kliknięcie przycisku **Options** i wybranie tego typu działania **Delete (usuwanie)** lub **Change (zmiana)** rys. (6.5b).



Rysunek 6.5: Modyfikacja serwerów. [źródło własne]

W przypadku modyfikacji danych dotyczących danej instancji platformy ponownie wysyłane jest zapytanie autoryzacyjne realizowane przez funkcję **X2MCheckConnection**. Wykorzystywane funkcje biblioteki: **X2MCheckConnection**.

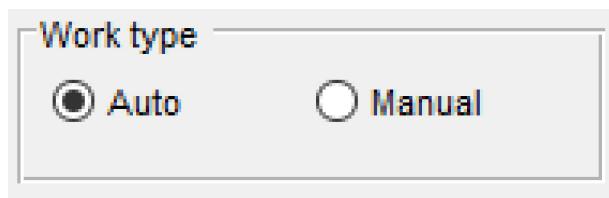
## 6.4 Query (zapytanie) - GUI

Zapytanie należy rozumieć jako odpytanie wszystkich skonfigurowanych instancji platformy XNAT (serwerów) o dany zasób, w tym przypadku pacjenta, i realizowany jest za pomocą funkcji. Możliwe do ustawienia parametry wyszukiwania to:

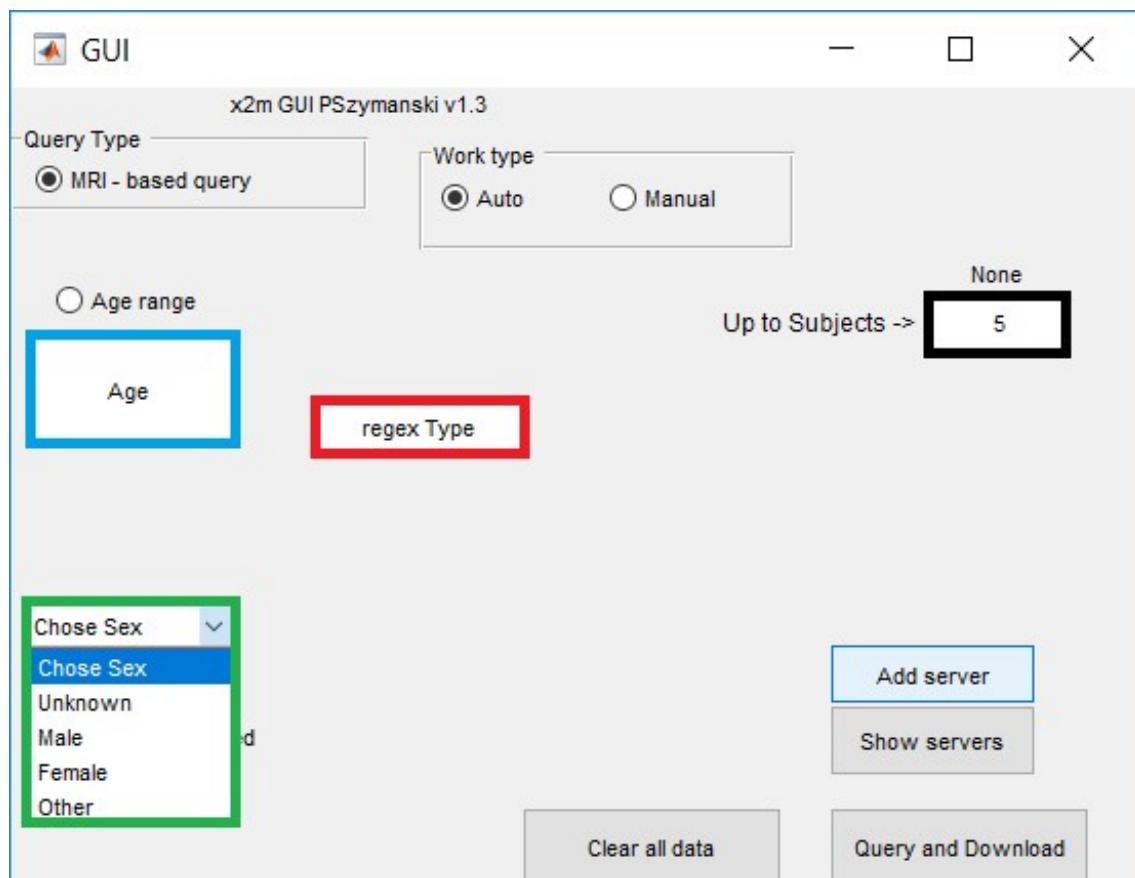
- 'Sex' - Płeć - zielony rys. (6.8a),
- 'upTo' - maksymalna liczba pobranych pacjentów czarny rys. (6.8a),
- 'Regex type' - szukany ciąg znaków w projekcie/badaniu na przykład 'fMRI' czerwony rys. (6.8a),
- 'Age' - dokładny wiek niebieski rys. (6.8a),
- 'Age From' - wiek od (zaznaczony przycisk 'Age range') fioletowy rys. (6.8b),
- 'Age to' - wiek do (zaznaczony przycisk 'Age range') niebieski rys. (6.8a).

Po ustaleniu wyżej wymienionych parametrów wyszukiwania możliwe są dwa tryby pracy **Auto** i **Manual** rys. (6.7).

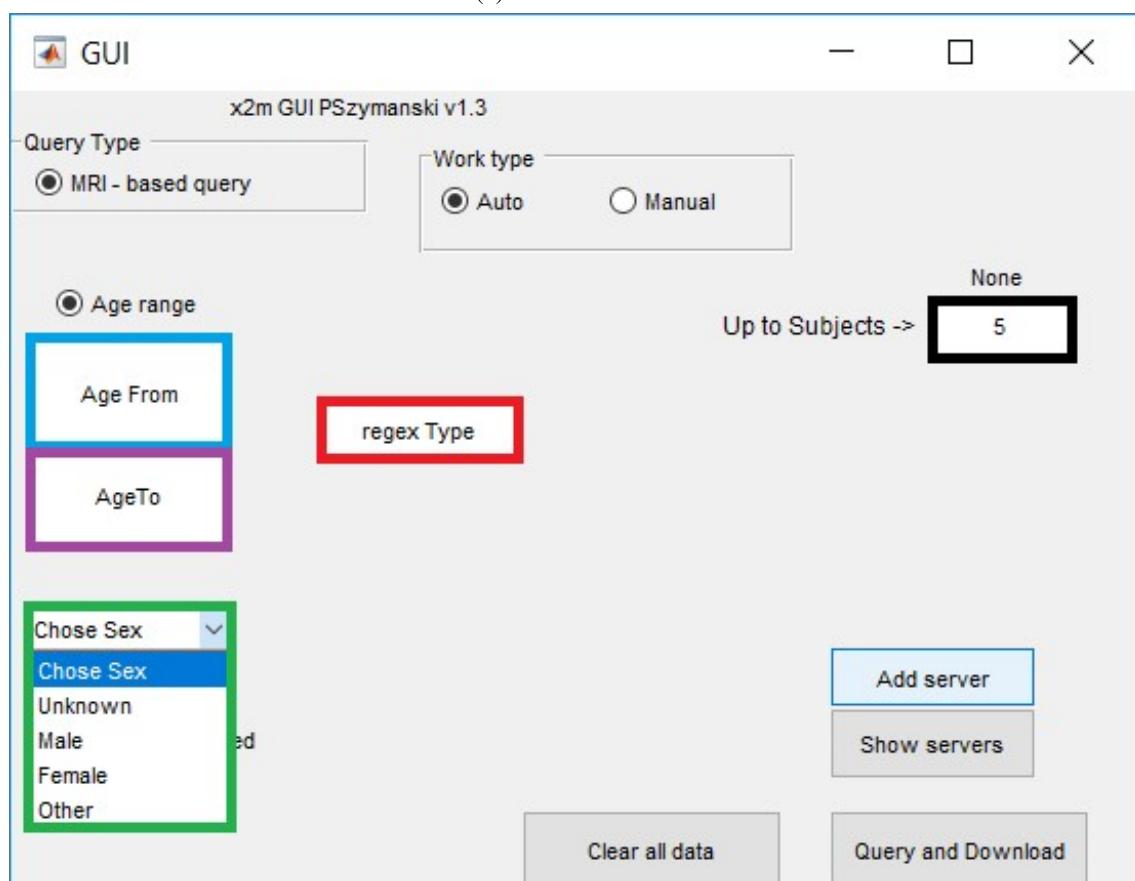
Tryby te wykonują również funkcje pobrania danych pacjenta, które zostaną opisane szczegółowo w kolejnych podrozdziałach.



Rysunek 6.6: Możliwe tryby pracy. [źródło własne]



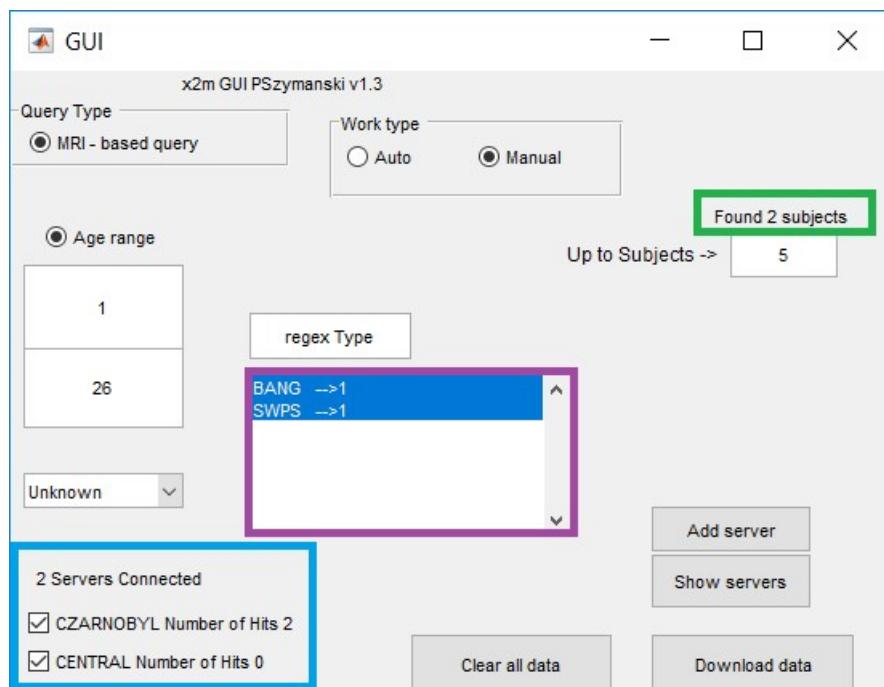
(a) Wiek dokładnie



(b) Zaznaczony przycisk 'Age range'

#### 6.4.1 Query - Manual

Tryb pracy manualnej udostępnia więcej możliwości niż tryb automatyczny, ale wymaga większej ingerencji użytkownika. W tym trybie po podaniu wyżej wymienionych parametrów i wcisnięciu przycisku **Query**, przedstawione zostaną użytkownikowi projekty wraz z liczbą pacjentów, zgodnych z wyszukiwanymi parametrami oraz liczba wyszukanych pacjentów z rozbiением na przeszukiwane serwery rys. (6.9). Proces ten jest odpowiedzialny za wyszukiwanie i jest realizowany przy użyciu funkcji **X2MQuery**. **Uwaga proces ten wymaga czasu** w przypadku, gdy maksymalna liczba pacjentów jest duża oraz gdy użytkownik połączony jest z Internetem za pomocą wolnego łącza (niska prędkość wyszukiwania i pobierania danych).



Rysunek 6.8: Query - Manual. Fioletowym kolorem zaznaczono liczbę pacjentów w ramach projektów, niebieskim liczbę pacjentów na serwerze, zielonym całkowitą liczbę pacjentów. [źródło własne]

Z rys. (6.9) widać, że łącznie znalezionych zostało dwóch pacjentów, obaj na serwerze *Czarnobyl*, jeden w ramach projektu *BANG*, drugi natomiast w ramach *SWPS*.

Jeśli istnieje potrzeba, aby pominąć dany serwer, w procesie pobierania danych należy odznaczyć checkbox, znajdujący się przed jego nazwą. Natomiast aby pominąć dany projekt, w procesie pobierania należy wciskając na klawiaturze komputera przycisk CTRL kliknąć na niego. Na rys. (6.9) widać, że żaden z projektów i serwerów nie zostanie pominięty w procesie pobierania.

Należy zaznaczyć, że przed wykonaniem procesu pobierania użytkownik może zmienić maksymalną liczbę pacjentów do pobrania. Po zaznaczeniu pożądanych serwerów, projektów i maksymalnej liczby pacjentów użytkownik może pobrać dane dotyczące pacjentów za pomocą przycisku **Download Data**. Dane pobierane są z wykorzystaniem funkcji **X2MDownloadData**. Podczas działania funkcji, użytkownik jest informowany na bieżąco, za pośrednictwem wiadomości w oknie poleceń środowiska Matlab, o:

- przeszukiwaniu danych pacjentów zielony rys. (6.10),
- błędach/sukcesie pobierania danych danego pacjenta i pozostały ilości danych do pobrania rys. (6.10),
- ominięciu pobierania wynikającego z podanego 'Regex type' rys. (6.10),
- dane których pacjentów zostały pobrane i pozostały ilości danych rys. (6.10).

```
Querying for detailed data of subject XNAT02_S00019
Querying for detailed data of subject XNAT02_S00139
Querying for detailed data of subject XNAT02_S00005
Querying for detailed data of subject XNAT02_S00010
Querying for detailed data of subject XNAT02_S00034
Querying for detailed data of subject XNAT02_S00035
Querying for detailed data of subject XNAT02_S00028
Querying for detailed data of subject XNAT02_S00030
Querying for detailed data of subject XNAT02_S00036
Querying for detailed data of subject XNAT02_S00022

Downloading data for subject XNAT02_S00019 omitted because of regex T1
Downloaded subject XNAT02_S00139 - completed 2 out of 10 subjects
Downloading data for subject XNAT02_S00005 omitted because of regex T1
Downloading data for subject XNAT02_S00010 omitted because of regex T1
Downloading data for subject XNAT02_S00034 omitted because of regex T1
Downloading data for subject XNAT02_S00035 omitted because of regex T1
Downloading data for subject XNAT02_S00028 omitted because of regex T1
Downloading data for subject XNAT02_S00030 omitted because of regex T1
Downloading data for subject XNAT02_S00036 omitted because of regex T1
Downloading data for subject XNAT02_S00022 omitted because of regex T1
```

Rysunek 6.9: Informacja o statusie pobierania danych. [źródło własne]

Dane pobierane są do katalogu wskazanego przez użytkownika w podrozdziale **Uruchamianie - GUI** (6.1). Dodatkowo w katalogu po zakończeniu całego procesu pojawi się log z przebiegiem działania całego procesu, to jest zbiór wszystkich wykonanych operacji do zakończenia pobrania opisany w podsekcji **X2MPrintLog** (5.2.1.4). Po zakończeniu jednego procesu użytkownik może zmienić kryteria wyszukiwania i analogicznie pobrać kolejny zestaw danych.

Wykorzystywane funkcje biblioteki:

**X2MQuery**,

**X2MDownloadData**.

#### **6.4.2 Query - Auto**

Automatyczny tryb pracy udostępnia co prawda mniej możliwości w porównaniu z poprzednio opisywanym, ale przy ograniczeniu do minimum ingerencji użytkownika. Dodatkowo użytkownik ma możliwość podania maila, na którego wysłana zostanie informacja o zakończeniu procesu pobierania. Najważniejszą różnicą jest to, że po podaniu opisanych w podrozdziale **Query (zapytanie) - GUI** (6.4) kryteriów wyszukiwana, w celu pobrania danych pacjentów należy jedynie wcisnąć przycisk **Query and Download**. Niestety ten tryb pracy nie pozwala wybierać projektów oraz serwerów, z których dane te mają zostać pobrane. Analogicznie jak do metody opisanej w sekcji **Query - Manual** (6.4.1), następuje zapis danych i logu do wskazanej w podrozdziale **Uruchomienie - GUI** (6.1) lokalizacji. Tryb ten zalecany jest do pobierania dużej ilości danych, ponieważ informuje mailowo użytkownika, kiedy proces się kończy. Problem czasu pobierania został przedstawiony w rozdziale **Weryfikacja działania** (7).

Wykorzystywane funkcje biblioteki:

**X2MQuery**,

**X2MDownloadData**.