**Warszawa, dn. 27.01.2010**

**Przemysław Ołtarzewski (199331)**

**Paweł Szczepański (199347)**

**Piotr Wieleba (199356)**

**Dokumentacja administracyjna do projektu z przedmiotu**

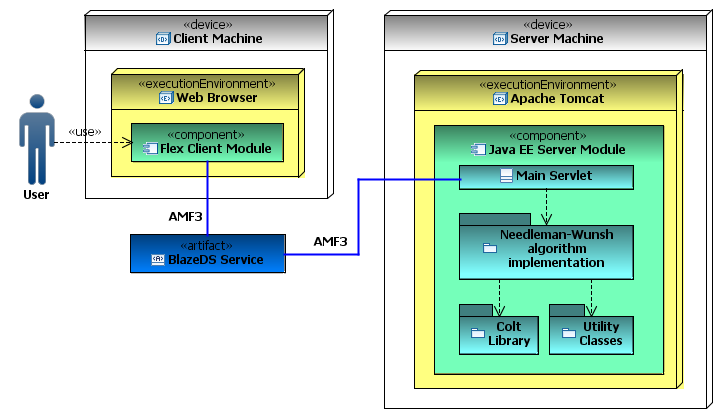
**Metody Bioinformatyki**

**Implementacja algorytmu Needlemana-Wunscha,**

**badającego podobieństwo dwu sekwencji**

Architektura rozwiązaniaArchitektura rozwiązania

Projekt architektury rozwiązania przedstawiono na diagramie zamieszczonym poniżej. Użytkownik komunikuje się z systemem przy wykorzystaniu modułu zrealizowanego w technologii [Adobe Flex](http://www.adobe.com/products/flex/), a uruchamianego poprzez przeglądarkę internetową. Plik SWF konieczny do wyświetlenia interfejsu użytkownika pobierany jest z serwera aplikacji i uruchamiany lokalnie na komputerze użytkownika.



Wsparciem interakcji z klientem po stronie Flexa zajmuje się mikro-architektura [Cairngorm](http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm). Wykorzystuje ona ogólnie przyjęte wzorce projektowe w celu uporządkowania komunikacji pomiędzy użytkownikiem, a kliencką stroną aplikacji. W katalogu dokumentacji znajduje się [plik SWF](cairngorm2_rpc.swf) obrazujący działanie [Cairngorma](http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm).

Moduł klienta komunikuje się z serwerem za pośrednictwem protokołu AMF3. Za serializację danych przekazywanych pomiędzy front-endem i back-endem odpowiada zestaw usług [Adobe BlazeDS](http://opensource.adobe.com/wiki/display/blazeds/BlazeDS/).

Rozwiązanie strukturalnie składa się z trzech osobnych projektów Mavena. Poniżej omówiono szczegółowo ich strukturę. W razie wątpliwości, należy przyjrzeć się odpowiednim plikom lub zasobom, dostarczonym w pakiecie zawierającym kod źródłowy rozwiązania.

# Projekt budujący Helix global aligner

GAL

|

|

+---GalBuild

build.bat

compile.bat

deploy.bat

execute.bat

pom.xml

README.txt

redeploy.bat

touch.exe

* **GalBuild** - katalog główny projektu
  + **build.bat** - plik wsadowy budujący kod Helix i osadzający w kontenerze Tomcat
  + **compile.bat** - plik wsadowy kompilujący kod Helix. Wywoływany przez build.bat
  + **deploy.bat** - plik wsadowy osadzający skompilowaną wersję w kontenerze Tomcat. Wywoływany przez build.bat
  + **execute.bat** - plik wsadowy wywołujący start kontenera Tomcat. Wymaga zmiany ścieżek na lokalizację, w której został zainstalowany Tomcat do poprawnego działania.
  + **pom.xml** - plik budujący Mavena, główny dla aplikacji.
  + **README.txt** - plik zawierający krótki opis programu Helix i jego przeznaczenia. Opis jest zbieżny z treścią przedstawioną użytkownikowi w widoku informacji
  + **redeploy.bat** - plik umożliwiający podmianę plików ‘na gorąco’ na działającej wersji Tomcata. Wykorzystywany do przyspieszonego projektowania i prototypowania. Wymaga podmiany ścieżek.
  + **touch.exe** - odpowiednik komendy touch systemu operacyjnego Linux. Umożliwia uaktualnianie daty pliku bez konieczności jego edycji - wykorzystywany przy szybkim prototypowaniu w celu interakcji z funkcjonalnością wprowadzania zmian na bieżąco przez kontener Tomcat.

# Projekt aplikacji klienckiej

GAL

|

|

+---GalClient

| | pom.xml

| |

| +---lib

| | cairngorm-2.2.1.swc

| | efflex-0.03.swc

| |

| \---src

| \---main

| +---flex

| | | empty.swf

| | | flekristal\_skin.css

| | | flekristal\_skin.swf

| | | flex\_skins.css

| | | flex\_skins.swf

| | | GalClient.mxml

| | |

| | \---gal

| | +---commands

| | | LoginCommand.as

| | | LogoutCommand.as

| | | RunSequenceCommand.as

| | |

| | +---control

| | | GalController.as

| | |

| | +---delegates

| | | LoginDelegate.as

| | | LogoutDelegate.as

| | | RunSequenceDelegate.as

| | |

| | +---dto

| | | RequestDTO.as

| | | ResultDTO.as

| | |

| | +---events

| | | LoginEvent.as

| | | LogoutEvent.as

| | | RunSequenceEvent.as

| | |

| | +---model

| | | | ConfusionMatrixRow.as

| | | | GalModelLocator.as

| | | |

| | | \---enums

| | | ActionViewStates.as

| | | MainViewStates.as

| | |

| | +---responders

| | | LoginResponder.as

| | | LogoutResponder.as

| | | RunSequenceResponder.as

| | |

| | +---service

| | | GalService.as

| | | GalServiceLocator.as

| | |

| | +---util

| | | IconManager.as

| | | ValidatorWrapper.as

| | |

| | \---view

| | | HeaderPanel.mxml

| | | InfoPanel.mxml

| | | InputPanel.mxml

| | | LoginPanel.mxml

| | | LoginView.mxml

| | | MainView.mxml

| | | OutputPanel.mxml

| | |

| | +---events

| | | RunEvent.as

| | |

| | +---renderers

| | | DnaRenderer.mxml

| | |

| | \---validators

| | ConfusionMatrixValidator.as

| | InputSequenceValidator.as

| |

| \---resources

| +---assets

| | +---fonts

| | | \---Chintzy

| | | chintzy.ttf

| | | chintzys.ttf

| | |

| | +---icons

| | | info\_blue\_48x48.png

| | | input\_blue\_48x48.png

| | | logout\_blue\_48x48.png

| | | next\_blue\_48x48.png

| | | output\_blue\_48x48.png

| | | previous\_blue\_48x48.png

| | | run\_blue\_48x48.png

| | |

| | \---images

| | adenine.svg

| | cytosine.svg

| | dnaOptimized.gif

| | dna\_chain\_darker.jpg

| | error.svg

| | gap.svg

| | guanine.svg

| | thymine.svg

| |

| \---template

| | AC\_OETags.js

| | index.template.html

| | playerProductInstall.swf

| |

| \---history

| history.css

| history.js

| historyFrame.html

|

* **GalClient** - katalog główny projektu klienta:
  + **pom.xml** - plik budujący Mavena. Zawiera polecenia kompilujące i pakujące kod klienta, zależności od zewnętrznych bibliotek.
* **lib** - katalog z bibliotekami zewnętrznymi wykorzystywanymi w projekcie. Biblioteki należy zainstalować w lokalnym repozytorium Mavena przed kompilacją:
  + **cairngorm-2.2.1.swc** - biblioteka Flex zawierająca klasy i interfejsy mikroarchitektury [Cairngorm](http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm);
  + **efflex-0.0.3.swc** - biblioteka Flex zawierająca klasy efektów, z których skorzystano w projekcie.
* **src/main/flex** - główny katalog źródłowy rozwiązania:
  + **empty.swf** - plik składowy skórki [Flekristal](http://fleksray.org/Flex_skin.html#Flekristal) zastosowanej w celu osiągnięcia wyglądu przyjaznego dla użytkownika, zawierający elementy 'puste' elementy, dla których nie zdefiniowano skóry;
  + **flekristal\_skin.css** - arkusz styli CSS dla Flex, służący do stylizowania wyglądu aplikacji - część Flekristal;
  + **flekristal\_skin.swf** - elementy Flex/Flash skóry Flekristal;
  + **flex\_skins.css** - kolejny arkusz styli Flekristal;
  + **flex\_skins.swf** - kolejna część elementów Flekristal;
  + **GalClient.mxml** - główny plik interfejsu użytkownika aplikacji klienckiej. Odpowiada między innymi za inicjalizację widoków i singletonów, dodanie styli, ustawienie adresu URL, z którym komunikuje się część kliencka na podstawie zmiennej przekazanej w powłoce HTML.
* **src/main/flex/gal/commands** - katalog pełniący rolę pakietu w sensie projektu Flex. Zawiera komendy w rozumieniu Cairngorm. Komendy reagują na zdarzenia biznesowe.
  + **LoginCommand.as** - plik ActionScript 3 definiujący klasę komendy logowania do części serwerowej. Reaguje na zdarzenie 'LoginEvent';
  + **LogoutCommand.as** - plik ActionScript 3 definiujący klasę komendy wylogowania z części serwerowej. Reaguje na zdarzenie biznesowe 'LogoutEvent';
  + **RunSequenceCommand.as** - plik AS 3 definiujący klasę komendy rozpoczęcia przetwarzania po stronie serwerowej. Reaguje na zdarzenie biznesowe 'RunSequenceEvent'.
* **src/main/flex/gal/delegates** - pakiet delegatów biznesowych w sensie Cairngorm. Delegaci odpowiadają za wykonanie logiki biznesowej w ramach komendy. Korzystają z usługi, modelu i responderów na potrzeby realizacji logiki:
  + **LoginDelegate.as** - plik AS 3 definiujący klasę zawierającą logikę komendy logowania do serwera;
  + **LogoutDelegate.as** - plik AS 3 definiujący klasę logiki komendy wylogowania z serwera;
  + **RunSequenceDelegate.as** - plik AS 3 definiujący klasę logiki komendy rozpoczęcia przetwarzania po stronie serwera.
* **src/main/flex/gal/control** - pakiet zawierający implementację frontowego kontrolera w rozumieniu Cairngorm:
  + **GalControllers.as** - plik AS 3 definiujący klasę frontowego kontrolera. Jest to singleton przechowujący mapowanie zdarzeń na komendy. Odpowiada za wykonanie odpowiedniej komendy, kiedy zostanie przekazane zdarzenie biznesowe.
* **src/main/flex/gal/delegates** - pakiet delegatów biznesowych w sensie Cairngorm. Delegaci odpowiadają za wykonanie logiki komend:
  + **LoginDelegate.as** - klasa AS 3 definiująca logikę dla komendy logowania;
  + **LogoutDelegate.as** - klasa AS 3 definiująca logikę dla komendy wylogowania;
  + **RunSequenceDelegate.as** - klasa AS 3 definiująca logikę dla komendy rozpoczęcia przetwarzania po stronie serwera.
* **src/main/flex/gal/dto** - pakiet obiektów transferu danych w sensie Flex / Cairngorm. Obiekty są serializowane do postaci binarnej przez BlazeDS, przesyłane przez protokół AMF3 i deserializowane do obiektów Javy po stronie serwera. Analogicznie funkcjonuje przesyłanie od strony serwera do klienta:
  + **RequestDTO.as** - klasa AS 3 zawierająca żądanie wykonania obliczeń przez backend. Zawiera sekwencje do dopasowania, macierz podobieństwa oraz karę za wstawienie przerwy;
  + **ResponseDTO.as** - klasa AS 3 zawierająca odpowiedź serwera po przeprowadzeniu obliczeń. Zawiera listę sekwencji DNA w postaci ciągów znaków. Sekwencje są ułożone naprzemiennie. Dwie sąsiadujące sekwencje tworzą jedno dopasowanie.
* **src/main/flex/gal/events** - pakiet zdarzeń biznesowych w sensie Cairngorm. Zdarzenie po utworzeniu i przekazaniu jest przechwytywane przez frontowy kontroler. Wykonywana jest związana z nim komenda.
  + **LoginEvent.as** - klasa AS 3 implementująca zdarzenie biznesowe, wywołujące komendę logowania do serwera. Przekazuje login i hasło wprowadzone przez użytkownika;
  + **LogoutEvent.as** - klasa AS 3 implementująca zdarzenie biznesowe, wywołujące komendę wylogowania z serwera;
  + **RunSequenceEvent.as** - klasa AS 3 implementująca zdarzenie biznesowe, wywołujące komendę przetwarzania danych.
* **src/main/flex/gal/model** - pakiet modelu danych dla Helix:
  + **ConfusionMatrixRow.as** - klasa AS 3 implementująca wiersz macierzy podobieństwa w algorytmie Needlemana-Wunscha. Ze względu na sposób działania komponentu DataGrid konieczne było utworzenie klasy modelującej wiersz, o unikalnych nazwach kolumn;
  + **GalModelLocator.as** - klasa AS 3 implementująca singleton lokalizatora modelu w sensie Cairngorm. Przechowuje dane wyświetlane w widokach aplikacji, zarówno wprowadzone przez użytkownika, jak również odebrane z serwera. Odpowiada za dostępność wszystkich danych w jednym miejscu aplikacji.
* **src/main/flex/gal/model/enums** - pakiet typów wyliczeniowych dla modelu danych Helix:
  + **ActionViewStates.as** - klasa AS 3 implementująca stałe oznaczające stany widoku akcji (widocznego w centralnej części ekranu pod panelem kontrolnym po zalogowaniu). Zawiera stany oznaczające wyświetlenie widoku wprowadzania danych, rezultatów, informacji o programie;
  + **MainViewStates.as** - klasa AS 3 implementujaca stałe oznaczające stany głównego widoku aplikacji - widocznego po uruchomieniu Helix. Modeluje dwa stany głównego widoku - logowania i wyświetlania widoku akcji.
* **src/main/flex/gal/responders** - pakiet klas responderów. Respondery są odpowiedzialne za reagowanie na asynchroniczną odpowiedź serwera wysyłaną po wykonaniu polecenia za pośrednictwem obiektu zdalnego wykorzystywanego przez usługę:
  + **LoginResponder.as** - klasa AS 3 implementująca zachowanie po otrzymaniu odpowiedzi na komendę logowania. W razie powodzenia przełącza aplikację do widoku akcji. Jeżeli wystąpiły problemy w czasie logowania, pozostawia wyświetlony ekran logowania i informuje użytkownika o problemie. Jest też odpowiedzialna za czyszczenie modelu przy udanej próbie zalogowania, aby nie pozostawiać danych z poprzedniej sesji w polach formularzy i kontrolkach - jest to dosyć częsty błąd wśród programistów Flex;
  + **LogoutResponder.as** - klasa AS 3 implementująca zachowanie po otrzymaniu odpowiedzi na komendę wylogowania. W razie powodzenia przełącza aplikację do widoku logowania. Przy niepowodzeniu, informuje dodatkowo użytkownika o problemie;
  + **RunSequenceResponder.as** - klasa AS 3 implementująca zachowanie po otrzymaniu odpowiedzi na komendę przeprowadzenia obliczeń. W przypadku powodzenia, przełącza aplikację z aktualnego widoku akcji do widoku rezultatów, uaktualniając dane wyświetlane użytkownikowi. W razie niepowodzenia, informuje o błędach przez wyświetlenie komunikatu.
* **src/flex/main/gal/service** - pakiet zawierający klasy związane z usługą komunikacji z częścią serwerową Helix:
  + **GalSevice.as** - klasa AS 3 zawierająca implementację usługi w postaci obiektu zdalnego. umożliwia tworzenie usługi, konfigurowanie adresu URL, z którym komunikuje się klient, definiowanie zestawu kanałów AMF po których przebiega komunikacja;
  + **GalServiceLocator.as** - klasa AS 3 zawierająca implementację lokalizatora dla usługi. Opakowuje usługę, włącza do struktury mikroarchitektury Cairngorm, zapewnia istnienie pojedynczej kopii w ramach aplikacji.
* **src/flex/main/gal/util** - pakiet klas użytkowych, wspomagających działanie aplikacji:
  + **IconManager.as** - klasa AS 3 odpowiedzialna za zarządzanie elementami wizualnymi - ikonami i obrazami. Wykorzystywana w tych widokach, które posługują się obrazami;
  + **ValidatorWrapper.as** - klasa AS 3 umożliwiająca jednoczesną walidację wielu okien i wykonywanie warunkowe metod, w zależności od wyniku walidacji.
* **src/flex/main/gal/view** - pakiet klas widoków aplikacji:
  + **HeaderPanel.mxml** - komponent MXML implementujący widok nagłówka. Zawiera przyciski służące w widoku głównym do kontrolowania aplikacji, interakcji użytkownika z serwerem;
  + **InfoPanel.mxml** - komponent MXML implementujący panel widoku informacji. Wyświetla krótki opis aplikacji Helix i jej funkcjonalności;
  + **InputPanel.mxml** - komponent MXML implementujący panel wprowadzania danych. Zawiera kontrolki macierzy podobieństwa, kary za przerwę, wprowadzania sekwencji, wizualizacji danych;
  + **LoginPanel.mxml** - komponent MXML implementujący panel logowania do aplikacji. Wyświetlany w momencie uruchamiania aplikacji, zapewnia autoryzowane korzystanie z funkcjonalności;
  + **LoginView.mxml** - komponent MXML służący do pozycjonowania panelu logowania w środku ekranu;
  + **MainView.mxml** - komponent MXML implementujący główny widok aplikacji. Zawiera w sobie panel nagłówka oraz zestaw paneli akcji, które są przełączane między sobą w zależności od interakcji użytkownika z klientem i serwerem Helix;
  + **OutputPanel.mxml** - komponent MXML implementujący widok rezultatów obliczeń dokonanych przez serwer. Umożliwia przełączanie się maksymalnie pomiędzy setką par dopasowań, śledzenie wizualne dopasowanych sekwencji, wyświetla moc dopasowania;
* **src/flex/main/gal/view/events** - pakiet wewnętrznych zdarzeń związanych z interakcją użytkownika z widokami klienta:
  + **RunEvent.as** - klasa AS 3 implementująca zdarzenie generowane, gdy użytkownik wybierze przycisk rozpoczęcia obliczeń z panelu nagłówka. Zdarzenie jest powiązane z metodą panelu wprowadzania danych, która pobiera dane z kontrolek, zapisuje w modelu i tworzy zdarzenie biznesowe zapoczątkowujące sekwencję przetwarzania po stronie klienta. Ciąg wygenerowanych zdarzeń kończy się wykonaniem algorytmu przez część serwerową dla przekazanych danych.
* **src/flex/main/gal/view/renderers** - pakiet zawierający komponenty graficzne służące do wypełniania innych widoków zgodnie z określonym wzorcem:
  + **DnaRenderer.mxml** - klasa AS 3 implementująca renderer zapewniający graficzne wyświetlanie sekwencji DNA w kontrolkach TileList w widoku wprowadzania danych i widoku rezultatów.
* **src/flex/main/gal/view/validators** - pakiet zawierający klasy walidujące dane wprowadzane przez użytkownika:
  + **ConfusionMatrixValidator.as** - klasa AS 3 odpowiedzialna za sprawdzanie poprawności danych wprowadzonych do macierzy podobieństwa. Reaguje wyświetlaniem podpowiedzi, gdy użytkownik wpisze do macierzy dane nie będące liczbą zmiennoprzecinkową;
  + **InputSequenceValidator.as** - klasa AS 3 odpowiedzialna za walidację poprawności wprowadzonej sekwencji DNA. Sekwencja może składać się jedynie z liter A,C,T,G.
* **src/main/resources** - główny katalog zasobów wykorzystywanych przez aplikację.
* **src/main/resources/assets/fonts** - katalog czcionek osadzonych w aplikacji. Czcionki zostały osadzone w arkuszach stylów.
* **src/main/resources/assets/fonts/Chintzy** - katalog czcionki Chintzy:
  + **chintzy.ttf** - czcionka wykorzystana w aplikacji - wersja pełna;
  + **chintzys.ttf** - czcionka wykorzystana w aplikacj - wersja 'szkieletowa' 3D.
* **/src/main/resources/assets/icons** - katalog ikon wykorzystanych w aplikacji:
  + **info\_blue\_48x48.png** - ikona przycisku panelu informacji;
  + **input\_blue\_48x48.png** - ikona przycisku panelu wprowadzania danych;
  + **logout\_blue\_48x48.png** - ikona przycisku wylogowywania z aplikacji;
  + **next\_blue\_48x48.png** - ikona przycisku następnego rezultatu w widoku wyników;
  + **output\_blue\_48x48.png** - ikona przycisku panelu wyników;
  + **previous\_blue\_48x48.png** - ikona przycisku poprzedniego rezultatu w widoku wyników;
  + **run\_blue\_48x48.png** - ikona przycisku wykonania obliczeń po stronie serwera.
* **/src/main/resources/images** - katalog obrazów wykorzystanych w aplikacji:
  + **adenine.svg** - obraz reprezentujący adeninę w widoku wprowadzania danych i widoku rezultatów;
  + **cytosine.svg** - obraz reprezentujący cytozynę w widoku wprowadzania danych i widoku rezultatów;
  + **dna\_chain\_darker.jpg** - obraz osadzony w głównym pliku aplikacji, stanowi tło dla Helix;
  + **error.svg** - obraz reprezentujący błąd w widoku wprowadzania danych i widoku rezultatów;
  + **gap.svg** - obraz reprezentujący przerwę łańcucha w widoku rezultatów;
  + **guanine.svg** - obraz reprezentujący guaninę w widoku wprowadzania danych i widoku rezultatów;
  + **thymine.svg** - obraz reprezentujący tyminę w widoku wprowadzania danych i widoku rezultatów.
* **/src/main/resources/template** - katalog szablonu html, na podstawie którego generowana jest strona html, w której osadzany jest plik swf powstały po skompilowaniu aplikacji klienta:
  + **AC\_OETags.js** - automatycznie utworzony plik javascript zawierający tagi pomocnicze do obsługi aplikacji Flexowej przez różne przeglądarki;
  + **index.template.html** - szablon głównego dokumentu (index.html), zawierającego osadzoną aplikację Helix. Jest odpowiedzialny za przekazanie przez parametry URLa, pod którym działa serwer. W wypadku braku odpowiedniej wersji flash playera, informuje o konieczności jej zainstalowania;
  + **playerProductInstall.swf** - umożliwia instalację wymaganego odtwarzacza Flash Player w przeglądarce.
* **/src/main/resources/template/history** - automatycznie wygenerowany katalog, zawierający pliki wspomagające obsługę historii przeglądarki dla aplikacji Flex - stosowany przy 'głębokim łączeniu' (deep linking):
  + **history.css** - plik styli ramki historii;
  + **history.js** - skrypty potrzebne do obsługi ramki historii;
  + **historyFrame.html** - główny kod ramki historii.

# Projekt aplikacji serwerowej

GAL

|

|

\---GalServer

| pom.xml

|

\---src

\---main

+---java

| \---gal

| +---application

| | GalService.java

| | GalServiceImpl.java

| |

| +---logging

| | BlazeSlf4jTarget.java

| |

| \---needleman

| \---wunsch

| FMatrixElement.java

| NeedlemanWunsch.java

| RequestDTO.java

| ResultDTO.java

|

+---resources

| logback.xml

|

\---webapp

+---META-INF

| MANIFEST.MF

|

\---WEB-INF

| dispatcher-servlet.xml

| touchfile

| web.xml

|

+---flex

| services-config.xml

|

\---spring

mainContext.xml

securityContext.xml

* **GalServer** - główny katalog projektu:
  + **pom.xml** - plik budujący Mavena dla części serwerowej. Zawiera wywołania i zależności pozwalające na kompilację i osadzenie całości aplikacji (łącznie z klientem) w kontenerze Tomcat.
* **src/main/java/gal** - główny katalog źródeł części serwerowej.
* **src/main/java/gal/application** - zawiera klasy implementujące usługę w rozumieniu Javy EE. Wykorzystywany jest mechanizm serwletów:
  + **GalService.java** - interfejs głównej usługi aplikacji. Zawiera sygnaturę metody odpowiedzialnej za wywołanie przetwarzania po stronie serwera;
  + **GalServiceImpl.java** - implementacja głównej usługia aplikacji. Zawiera metodę wywoływaną zdalnie przez klienta. Odpowiada za wywołanie algorytmu Needlemana-Wunscha i początek przetwarzania.
* **src/main/java/gal/application/logging** - pakiet Javy zawierający klasy odpowiedzialne za logowanie informacji do plików w czasie działania aplikacji. Informacje zapisywane są w podkatalogu logs Tomcata, w pliku gal.log:
  + **BlazeSlf4jTarget.java** - implementacja obiektu docelowego fasady logowania Slf4j. Opakowuje ona inne systemy logowania i pozwala zebrać dane z różnych części aplikacji w jednym pliku, śledzić synchronizowane działanie poszczególnych komponentów. Klasa implementuje przechwytywanie informacji z systemu logowania BlazeDS do loggera Slf4j.
* **src/main/needleman/wunsch** - główne klasy implementujące algorytm Needlemana-Wunscha:
  + **FMatrixElement.java** – klasa reprezentująca pojedynczy element macierzy „F”. Przechowuje informacje o wartości współczynnika podobieństwa w danym miejscu oraz informację od których sąsiadów (lewo, góra, lewy-górny skos) można po przeliczeniu uzyskać tą wartość;
  + **NeedlemanWunsch.java** – klasa służąca do obsługi algorytmu Needlemana-Wunsha. Posiada konstruktor wczytujący dane z obiektu klasy RequestDTO służący inicjalizacji podstawowych danych wejściowych (sekwencje do porównania, macierz podobieństwa oraz kara za przerwę). Uruchomienie algorytmu skutkuje utworzeniem macierzy „F” oraz obiektu klasy ResultDTO zawierającego wynik działania algorytmu (współczynnik podobieństwa obu sekwencji oraz do 100 przykładów ich najlepszego dopasowania);
  + **RequestDTO.java** – klasa służąca przesłaniu informacji wprowadzonych przez użytkownika systemu do serwera w celu dostarczenia danych potrzebnych do uruchomienia algorytmu. Zawiera obie sekwencje do porównania, macierz podobieństwa oraz karę za przerwę. Obiekty tej klasy są serializowane przez BlazeDS;
  + **ResultDTO.java** – klasa służąca odesłaniu wyniku wygenerowanego po stronie serwera aplikacji z powrotem do przeglądarki po stronie klienckiej. Obiekt tej klasy zawiera współczynnik podobieństwa obu sekwencji oraz maksymalnie 100 przykładów (a przynajmniej jeden) dopasowania obu sekwencji dla danej wartości współczynnika dopasowania.
* **src/main/resources** - główny katalog zasobów po stronie serwerowej Helix:
  + **logback.xml** - plik konfiguracyjny systemu logowania Logback - następcy Log4j o rozszerzonych możliwościach. W projekcie, jest on wykorzystywany za pośrednictwem fasady Slf4j.
* **src/main/webapp** - katalog zawierający konfigurację potrzebną do uruchomienia aplikacji sieciowej w kontenerze Tomcat, jak i do osadzenia aplikacji w postaci artefaktu w repozytorium Marena.
* **src/main/webapp/META-INF** - katalog meta-informacji Marena:
  + **MANIFEST.MF** - plik zawierający dane o nazwie, identyfikatorze grupy oraz wersji artefaktu Mavena, jakim z punktu widzenia tego systemu jest Helix.
* **src/main/webapp/WEB-INF** - katalog wykorzystywany przez Tomcata do uruchomienia aplikacji Helix - zawiera konfigurację ustawień aplikacji Java EE:
  + **dispatcher-servlet.xml** - plik konfiguracyjny serwletu pośredniczącego (klasa DispatcherServlet frameworku Spring). Serwlet przekazuje wywołania klientów do skonfigurowanych w pliku serwletów-dzieci, umożliwia mapowanie ścieżek URL na konkretne serwety;
  + **touchfile** - plik wykorzystywany przy automatycznym uaktualnianiu wersji aplikacji osadzonej w kontenerze Tomcat. Wymagana jest specjalna konfiguracja. Szczegóły w dokumentacji Tomcat. Po wgraniu pliku do katalogu WEB-INF rozpakowanego archiwum w podkatalogu webapps Tomcata, kontener wyłącza i ponownie osadza aplikację, wprowadzając zmiany na bieżąco. Wykorzystywany przy szybkim prototypowaniu.
* **src/main/webapp/WEB-INF/flex** - podkatalog zawierający konfigurację usług BlazeDS, z którymi komunikuje się klient Flexowy za pośrednictwem kanału AMF:
  + **services-config.xml** - plik zawiera konfigurację udostępnionej usługi, do której żądania routowane są przez serwlet pośredniczący. Zawiera dodatkowe tagi projektu Spring BlazeDS integration, umożliwiające między innymi integrację z mechanizmami bezpieczeństwa wprowadzonymi przy logowaniu użytkownika.
* **src/main/webapp/WEB-INF/spring** - podkatalog konfiguracji frameworku Spring:
  + **mainContext.xml** - plik konfigurujący główny kontekst aplikacji w rozumieniu Javy EE i Spring. Kontekst zawiera informacje potrzebne do implementacji wzorca IoC (inversion of control) i DI (dependency injection) Springa. Zawiera definicje obiektów tworzonych przy starcie serwera i powiązań między nimi;
  + **securityContext.xml** - plik konfiguracyjny Spring Security. Określa, jakie zabezpieczenia są wykorzystywane przy potwierdzaniu tożsamości i autoryzacji użytkownika do przeprowadzania konkretnych operacji. W Helix zastosowano bezpieczeństwo oparte na rolach osadzonych w pliku konfiguracyjnym. Jest to najprostszy schemat, który może jednak zostać odpowiednio rozbudowany.