NETIT - Network Emulation and Topology Implementation Tool

Projekt NETIT został zaprojektowany z myślą o natywnej wieloplatformowości, co osiągnięto poprzez implementację w języku Java oraz zastosowanie biblioteki JavaFX do obsługi interfejsu graficznego. Na dzień 2 lipca 2025 roku aplikacja działa w środowisku wykonawczym opartym na OpenJDK w wersji 21.0.7, wydanym 15 kwietnia 2025. Jest to wydanie dystrybuowane przez Red Hat (`Red_Hat-21.0.7.0.6-2`), zawierające zarówno środowisko wykonawcze (OpenJDK Runtime Environment), jak i maszynę wirtualną (OpenJDK 64-Bit Server VM), działającą w trybie „mixed mode" z aktywnym mechanizmem dzielenia danych klas (`class data sharing`), który umożliwia skrócenie czasu uruchamiania aplikacji i redukcję zużycia pamięci przez współdzielenie wcześniej przygotowanych metadanych klas.

OpenJDK 21 to wersja LTS (Long-Term Support), co oznacza, że posiada dłuższy cykl wsparcia technicznego oraz aktualizacji bezpieczeństwa, co jest szczególnie istotne w kontekście projektów produkcyjnych i długoterminowych wdrożeń. Wersja 21.0.7 zawiera zbiorczą paczkę poprawek błędów, aktualizacji zabezpieczeń oraz optymalizacji wydajności, zgodną z harmonogramem kwartalnych wydań aktualizacyjnych JDK.

Warstwa graficzna aplikacji NETIT została zaimplementowana przy użyciu biblioteki OpenJFX w wersji 21.0.7, będącej oficjalnym odpowiednikiem JavaFX dla JDK 21. JavaFX pełni rolę samodzielnej biblioteki GUI, oferującej bogaty zestaw komponentów interfejsu użytkownika, mechanizmy reaktywne (Observable API), wsparcie dla CSS i FXML (język deklaratywny do opisu interfejsów), jak również akcelerację sprzętową z wykorzystaniem Direct3D lub OpenGL, w zależności od systemu operacyjnego i dostępnych sterowników graficznych.

Zestawienie OpenJDK 21.0.7 z JavaFX 21.0.7 zapewnia pełną zgodność binarną, stabilność oraz możliwość wykorzystania nowoczesnych konstrukcji językowych wprowadzonych w kolejnych wersjach Javy, takich jak rekordy, wzorce dopasowania (`pattern matching`), lokalne klasy metodowe, czy zaktualizowany system inkapsulacji modułów (JPMS – Java Platform Module System). Dzięki temu środowisko uruchomieniowe NETIT jest jednocześnie nowoczesne, zgodne z aktualnymi standardami technicznymi i gotowe do dalszego rozwoju w ramach ekosystemu JVM.

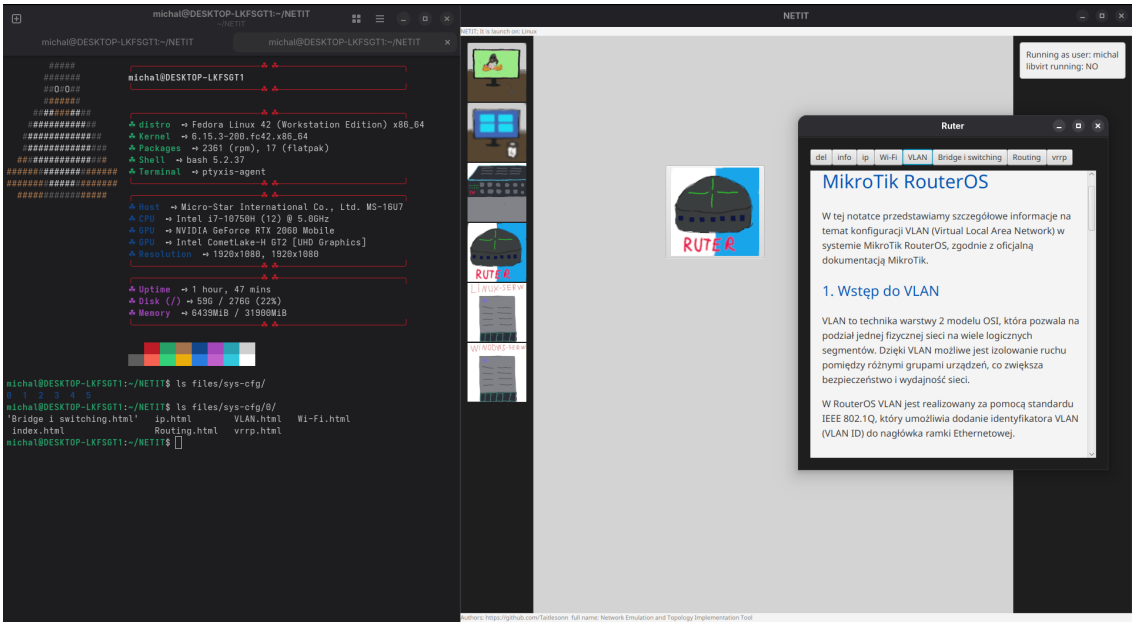
W celu uproszczenia procesu integracji zewnętrznych rozszerzeń tworzonych przez społeczność, projekt NETIT został zaprojektowany w sposób umożliwiający łatwe dodawanie i modyfikowanie elementów interaktywnych związanych z topologią sieci bez konieczności ingerencji w kod źródłowy aplikacji. Mechanizm ten opiera się na użyciu plików HTML jako interfejsów wizualnych dla poszczególnych obiektów wirtualnej topologii (takich jak routery, switche, systemy operacyjne Windows, Linux, Windows Server, Linux Server itd.).

Każdy obiekt obecny w topologii ma przypisaną strukturę zakładek, które są reprezentowane jako samodzielne pliki `.html`. Pliki te są umieszczane w odpowiednich podkatalogach lokalizacji `files/sys-cfg/`, w zależności od typu i identyfikatora obiektu. Przykładowo,

`files/sys-cfg/0/` może zawierać pliki HTML odpowiadające pierwszemu obiektowi, `files/sys-cfg/1/` drugiemu itd. Każdy z tych katalogów może zawierać dowolną liczbę plików HTML, z których każdy reprezentuje jedną zakładkę funkcjonalną w interfejsie użytkownika.

Interaktywność i renderowanie zakładek w aplikacji NETIT realizowane są przy użyciu komponentu `javafx.scene.web.WebView`, który stanowi osadzoną przeglądarkę opartą na silniku WebKit. Po kliknięciu na dany obiekt w wizualnej topologii, uruchamiany jest komponent drugorzędnego interfejsu użytkownika (Secondary UI), który dynamicznie ładuje i interpretuje zawartość wszystkich dostępnych plików HTML przypisanych do danego obiektu. Proces ten jest całkowicie automatyczny – nie wymaga żadnej dodatkowej konfiguracji ani rejestracji plików – wystarczy ich obecność w odpowiednim folderze.

Dzięki takiemu podejściu rozszerzalność aplikacji staje się naturalna i intuicyjna. Użytkownicy oraz członkowie społeczności mogą tworzyć własne panele konfiguracyjne, formularze, instrukcje lub dokumentację, zapisując je jako zwykłe pliki HTML i umieszczając w katalogach struktury `files/sys-cfg/`. Aplikacja samodzielnie wykrywa i prezentuje je jako funkcjonalne zakładki, co znacząco obniża próg wejścia dla użytkowników mniej zaawansowanych programistycznie i pozwala na szybkie tworzenie spersonalizowanych rozwiązań w oparciu o standardowe technologie webowe.



W projekcie NETIT znajduje się gotowy do instalacji pakiet RPM przeznaczony dla systemów zgodnych z Red Hat (np. Fedora, CentOS, RHEL, openSUSE). Instalator ma na celu uproszczenie wdrażania aplikacji użytkownikom końcowym, którzy korzystają z systemów wspierających menedżera pakietów `rpm`. Choć na obecnym etapie wdrażanie odbywa się manualnie, bez wykorzystania publicznego repozytorium `rpmbuild` lub `dnf/yum/zypper`, struktura pakietu została przygotowana w pełni zgodnie z konwencją RPM.

Plik `netit.spec` definiuje sposób budowania i instalacji pakietu. Kluczowe informacje zawarte w specyfikacji to:

- Name: `netit` – nazwa pakietu.
- Version: `0.3` – aktualna wersja aplikacji.
- Release: `1%{?dist}` – numer wydania, z opcjonalną dystrybucją.
- Summary: krótki opis pakietu.
- License: `ISC` – bardzo liberalna licencja open source.
- Source0: archiwum źródłowe (`netit-0.3.tar.gz`) zawierające gotowe pliki aplikacji.
- BuildArch: `x86_64` – pakiet przeznaczony wyłącznie dla 64-bitowych systemów.
- Requires: `java-21-openjdk` – aplikacja wymaga zainstalowanego środowiska OpenJDK 21.

W sekcji `%build` brak jest instrukcji budowania, ponieważ aplikacja została wcześniej skompilowana i zawiera już plik `netit.jar`, skompilowany katalog `out/` oraz skopiowany SDK JavaFX. RPM nie wykonuje więc kompilacji, a jedynie instaluje pliki binarne w odpowiednie lokalizacje systemowe.

W sekcji `%install` określono dwie główne ścieżki instalacji:

- `/opt/netit/` – tutaj trafiają wszystkie pliki aplikacji: JAR, zasoby, oraz SDK JavaFX.
- `/usr/bin/netit` – skrypt uruchamiający, który zapewnia użytkownikowi możliwość uruchomienia aplikacji poleceniem `netit` z poziomu terminala. Skrypt ustawia zmienną `LIBGL_ALWAYS_SOFTWARE=1` oraz wymusza software'owe renderowanie (`-Dprism.order=sw`), co zwiększa kompatybilność z maszynami nieposiadającymi odpowiednich sterowników GPU.

Do uruchomienia aplikacji wykorzystywana jest maszyna wirtualna Javy z dodanymi modułami `javafx.controls`, `javafx.fxml` oraz `javafx.web`, a ścieżka do bibliotek JavaFX (`--module-path`) wskazuje na lokalne SDK dostarczone w `/opt/netit/javafx-sdk-21.0.7/lib`.

Sekcja `%files` deklaruje, które pliki są instalowane przez pakiet: cały katalog `/opt/netit` oraz skrypt uruchamiający w `/usr/bin/netit`.

Na chwilę obecną wdrażanie pakietu RPM odbywa się ręcznie – użytkownik pobiera plik `.rpm` i instaluje go poleceniem `sudo rpm -i netit-0.3-1.x86_64.rpm` lub `sudo dnf install ./netit-0.3-1.x86_64.rpm`. Projekt nie korzysta jeszcze z systemu budowania RPM typu `rpmbuild` ani z infrastruktury repozytorium, ale struktura pakietu została przygotowana w sposób pozwalający na łatwą integrację z takim systemem w przyszłości.

Proces kompilacji i budowy aplikacji NETIT został zaprojektowany z myślą o prostocie, przewidywalności oraz maksymalnej kontroli nad środowiskiem wykonawczym. Projekt bazuje na systemie budowania Maven, ale nie wykorzystuje zależności z repozytoriów zewnętrznych – wszystkie niezbędne biblioteki (w szczególności JavaFX) są dostarczane lokalnie i wskazywane jawnie poprzez ścieżki bezwzględne. Dzięki temu proces jest w pełni deterministyczny i nie zależy od sieci ani wersji bibliotek zewnętrznych.

Centralnym elementem budowy jest plik `pom.xml`, który definiuje strukturę projektu Maven, ustawienia kompilatora oraz używane wtyczki. Aplikacja NETIT jest kompilowana z wykorzystaniem JDK 21, co znajduje odzwierciedlenie zarówno w parametrach `<maven.compiler.source>` i `<maven.compiler.target>`, jak i w konfiguracji `maven-compiler-plugin`, który dodatkowo ustawia ścieżkę do bibliotek JavaFX za pomocą opcji `--module-path`. W pliku zadeklarowano również dodanie modułów `javafx.controls` i `javafx.fxml`, a przy użyciu `exec-maven-plugin` oraz `maven-shade-plugin` zapewniono możliwość uruchamiania oraz spakowania aplikacji w sposób zgodny z wymaganiami modularnej Javy (JPMS).

Proces budowania został zautomatyzowany za pomocą skryptu `build.sh`, który pełni funkcję zarówno wrappera Mavena, jak i narzędzia uruchomieniowego. Skrypt ten umożliwia czyszczenie wcześniejszych artefaktów (`--clean`), buduje projekt za pomocą polecenia `mvn clean package`, a następnie przygotowuje katalog wyjściowy `out/`, do którego kopiowany jest wynikowy plik JAR (`netit.jar`) oraz folder `files/`, zawierający wszystkie dodatkowe zasoby konfiguracyjne aplikacji (np. pliki HTML interfejsu, grafiki, struktury topologii).

Ostatnia faza działania skryptu polega na bezpośrednim uruchomieniu aplikacji za pomocą polecenia `java`, z odpowiednio skonfigurowanym `--module-path`, który wskazuje na folder `lib` w SDK JavaFX. Dołączone zostają również niezbędne moduły JavaFX: `javafx.controls`, `javafx.fxml` oraz `javafx.web` – ten ostatni jest szczególnie istotny, ponieważ aplikacja NETIT wykorzystuje komponent `WebView` do renderowania dynamicznych elementów GUI opartych o HTML.

Aplikacja jest kompilowana jako modularny artefakt, gdzie główną klasą startową jest `com.netit.Main`. Struktura pakietowania wykorzystuje `maven-shade-plugin` do utworzenia tzw. "fat JAR-a" (czyli JAR-a zawierającego wszystkie wymagane klasy), dzięki czemu aplikacja może być łatwo przenoszona i uruchamiana bez potrzeby dołączania bibliotek zewnętrznych.

Co ważne, proces ten zakłada, że użytkownik posiada lokalnie zainstalowane SDK JavaFX w wersji 21.0.7, domyślnie dostępne pod ścieżką `/opt/javafx-sdk-21/javafx-sdk-21.0.7/lib`. W razie potrzeby, ścieżka ta może być nadpisana za pomocą zmiennej środowiskowej `JAVA_FX_SDK`. Podobnie, dodatkowe opcje dla Mavena i Javy można podać poprzez zmienne `MAVEN_ARGS` oraz `JAVA_ARGS`, co pozwala na dużą elastyczność bez modyfikacji samego skryptu.

Podsumowując, proces budowy NETIT został w całości zoptymalizowany pod kątem spójności i kontroli. Pomija automatyczne pobieranie zależności z internetu, na rzecz lokalnych, jawnie zarządzanych komponentów. To podejście minimalizuje ryzyko niespójności wersji i upraszcza wdrożenia w środowiskach izolowanych lub bez dostępu do internetu.