

6. Realizacja systemu

Przed rozpoczęciem prac nad systemem konieczna była decyzja z jakich narzędzi skorzystać przy tworzeniu aplikacji. Program musiał być kompatybilny z oprogramowaniem obecnym na sprzęcie w laboratorium, oraz nie mógł polegać na zewnętrznych zasobach które trzeba by instalować na maszynach oraz regularnie aktualizować. Ważnym aspektem była też wydajność, aby symulacja mogła odbywać się w możliwie krótkim czasie nawet przy dużej ilości danych wejściowych. Aplikacja powinna być również stabilna, aby zapewnić możliwość bezproblemowego korzystania z niej podczas trwania laboratorium.

Oprócz wymagań co do funkcjonowania finalnej aplikacji, musiałem też zwrócić uwagę na stopień skomplikowania tworzenia zaawansowanego interfejsu użytkownika w różnych językach programowania. Za zaletę postrzegana była łatwość wprowadzania zmian oraz utrzymania kodu, zarówno jak jego przejrzystość. Wybrane narzędzie musiało również umożliwiać pełną kontrolę nad tworzonymi elementami interfejsu, aby móc implementować własną logikę dynamicznego pojawiania się odpowiednich przycisków oraz pól tekstowych.

6.1 Język programowania

Podstawowym wyborem którego musiałem dokonać był wybór języka programowania. Większość wiodących języków udostępnia biblioteki służące tworzeniu zaawansowanych interfejsów użytkownika, dlatego przeanalizowałem wiele z popularnych narzędzi i zdecydowałem się na skorzystanie z Javy. Poniżej przedstawię niektóre z możliwości jak również ich wady i zalety.

6.1.1 Python

Język ten został zapoczątkowany w latach dziewięćdziesiątych, jednak popularność zaczął szybko zyskiwać dopiero w roku 2004. Aktualnie jest jednym z najpopularniejszych języków programowania ogólnego zastosowania, a liczba korzystających z niego osób wciąż stale rośnie.

Głównymi zaletami Pythona jest przejrzystość i prostota kodu, jak również jego ekspresywność. Jedna linia napisana w Pythonie jest równoważna nawet do 6 linii w języku C. Dzięki temu aplikacje powstają szybko i efektywnie, a sam kod jest czytelny i zwięzły. Język ten jest również częściowo interpretowany co przekłada się na brak dodatkowych wymagań przy korzystaniu z niego na wielu różnych systemach operacyjnych.

Główną wadą jest brak uznanego i wspieranego narzędzia do tworzenia interfejsów użytkownika. Istnieje wiele bibliotek które umożliwiają tworzenie różnych elementów interaktywnych, jednak większość z nich wywołuje się z poziomu kodu, co przy tworzeniu dużej liczby paneli i pól tekstowych staje się wyjątkowo czasochłonne i wymaga dużej uwagi. Aby uruchomić skrypt Pythona na komputerze, wymagana jest uprzednia instalacja odpowiedniego oprogramowania oraz zalecana jest jego regularna aktualizacja, co mogłoby stwarzać problemy na sprzęcie laboratoryjnym.

6.1.2 C++

Wywodzący się z języka C, stworzony w latach osiemdziesiątych język C++ jest najlepszym wyborem pod względem wydajności i stabilności stworzonego kodu. Stanowi on swojego rodzaju standard w programowaniu przez co dostępne zasoby są zawsze na najwyższym poziomie. W języku tym stosowana jest jednoznaczna składnia z silną kontrolą typów, co pozwala na bardzo szybkie i automatyczne modyfikowanie dużych bloków logiki programu.

Powodem dla którego nie wybrałem tego języka jest jego mocna zależność od obecnych na komputerze bibliotek i wymóg uprzedniej kompilacji programu. Przez to nie jest on łatwo przenośny pomiędzy różnymi systemami operacyjnymi. Ponieważ C++ daje kontrolę programiście nad zarządzaniem pamięcią, tworzenie aplikacji wymaga wyjątkowej uwagi oraz znajomości wykorzystywanych bibliotek. Podobnie do Pythona, tutaj również nie ma uznanego narzędzia do łatwego tworzenia i konfigurowania interfejsów.

6.1.3 Java

Jest to język stworzony w roku 1991. Bardzo szybko zyskał on na popularności, dzięki czemu aktualnie jest najczęściej używanym językiem programowania na świecie. Wynik ten jest skutkiem łatwości uruchamiania aplikacji na różnych systemach operacyjnych, w tym na urządzeniach mobilnych, jak również jednoznaczna struktura kodu oraz reguły zaprojektowane z myślą o automatycznej kontroli możliwych błędów programistycznych.

Największym czynnikiem decydującym o wyborze tego właśnie języka jest dostępność narzędzi do tworzenia aplikacji okienkowych. Z powodu ogromnej popularności dostępnych jest wiele gotowych programów do tworzenia elementów interfejsu, jak również większość problemów które można napotkać jest szczegółowo omówiona na forach internetowych. Dzięki temu umożliwienie użytkownikowi prostego wprowadzania parametrów wejściowych oraz analizowania wyników było znacznie uproszczone, przez co więcej czasu mogłem poświęcić na dopracowanie logiki aplikacji oraz testy.

Jedną z wad Javy która mogła negatywnie wpłynąć na działanie aplikacji jest mniejsza w porównaniu do C++ wydajność. Przez to symulacja przebiega wolniej, jednak różnica jest niezauważalna nawet przy znacznej liczbie maszyn oraz zadań w systemie.

6.2 Środowisko programistyczne

Po zdecydowaniu się na wykorzystanie Javy jako języka do napisania aplikacji, musiałem wybrać odpowiednie środowisko do utworzenia w nim projektu z kodem programu. Spośród wielu popularnych możliwości przeanalizowałem dwa najpopularniejsze środowiska: Eclipse oraz NetBeans, decydując się na wykorzystanie drugiego.

6.2.1 Eclipse

Jest to platforma napisana w 2004 roku. Jej główną zaletą jest rozbudowana biblioteka dostępnych rozszerzeń które dostosowują środowisko do indywidualnych potrzeb użytkownika. Narzędzie to w znacznym stopniu ułatwia zarządzanie dużymi projektami informatycznymi oraz ma wiele wbudowanych funkcjonalności. Główną wadą tego oprogramowania jest jego złożoność. Początkujący użytkownik ma dostęp do wszystkich zaawansowanych funkcji programu przez co wykonywanie prostych czynności może być utrudnione.

6.2.2 NetBeans

Środowisko to ma swoje początki jako projekt studencki zrealizowany w 1996 roku na Uniwersytecie Karola w Pradze. W 2010 roku projekt przeszedł na własność firmy Oracle, co zapoczątkowało gwałtowny wzrost jego popularności.

Najbardziej wartościową dla mnie cechą tego oprogramowania jest wbudowany w niego edytor interfejsu użytkownika. Umożliwia on proste tworzenie elementów interaktywnych poprzez wybieranie ich ze specjalnego zasobnika. Kod odpowiadający za komunikację przycisków i pól tekstowych z logiką programu generowany jest automatycznie, co redukuje czas potrzebny na pisanie programu oraz zmniejsza liczbę potencjalnych błędów. Narzędzie to jest kompaktowe oraz przyjazne użytkownikowi. Wszystkie jego funkcjonalności są łatwo i intuicyjnie dostępne.

6.3 Kontrola wersji

Podczas rozwijania każdego projektu informatycznego niezbędne jest korzystanie z oprogramowania zapewniającego kopie bezpieczeństwa tworzonego kodu. W razie wprowadzenia funkcjonalności powodujących nieprawidłowe działanie programu należy mieć możliwość szybkiego cofnięcia się do wersji stabilnej, jak również analizy wprowadzonych błędów w celu ich naprawy. Problemy te napotykanne są często przy tworzeniu komercyjnych aplikacji, dlatego powstało wiele narzędzi ułatwiających prowadzenie kontroli wersji.

6.3.1 GIT

Jest to system początkowo stworzony jako narzędzie wspomagające rozwój jądra Linux. Jego główną zaletą jest zapewnienie wsparcia dla rozgałęzionych procesów tworzenia oprogramowania, co pozwala na równoczesną pracę wielu osób nad systemem, lub równoległe rozwijanie wielu jego funkcjonalności. Dzięki temu jest on aktualnie najpopularniejszym systemem kontroli wersji na świecie.

Podczas pisania aplikacji, GIT stanowił miejsce bezpiecznego przechowywania kopii zapasowych, jak również udostępniał historię zmian projektu, co pozwalało na szybką

analizę błędów polegających na niepoprawnie współpracujących ze sobą części programu. W celu zabezpieczenia się przed awarią komputera lub dysku twardego, repozytorium projektu utworzyłem w darmowym serwisie internetowym GitHub. Dzięki temu oprócz lokalnych kopii zapasowych istniała też kopia zewnętrzna projektu, która zapewniała brak możliwości utracenia postępu prac.

Jako narzędzie wspomagające korzystanie z wybranego systemu kontroli wersji wykorzystałem program Git Extensions. Komunikacja z serwerem repozytorium odbywa się za pomocą poleceń tekstowych wywoływanych z konsoli, dlatego użycie jednego z wielu narzędzi udostępniających graficzny interfejs użytkownika znacznie skraca czas wymagany na odpowiednie tworzenie kopii projektu jak również poruszanie się po jego rozgałęzieniach.