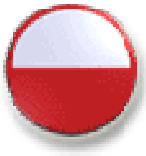


Polsko-Japońska Wyższa Szkoła Technik Komputerowych

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Tomasz Paweł Skrobol

Nr albumu: 4885

Generyczny system do tworzenia i obsługi formularzy działający przez www.

Praca magisterska
napisana pod kierunkiem
dr inż. Mariusza Trzaski

Warszawa, lipiec 2009

STRESZCZENIE

Celem niniejszej pracy jest opracowanie zasad tworzenia oprogramowania spełniającego wymagania stawiane przed aplikacjami wykorzystywanymi w dużych przedsiębiorstwach. Zaprezentowane zostaną możliwości technologiczne modularyzacji oprogramowania oraz wykorzystania technik pozwalających zwiększyć wydajność systemu na nim opartego. Przykładem takiej aplikacji będzie tworzenie i wypełnianie formularzy. Przedstawiony problem dotyczy aplikacji klasy Enterprise działających na serwerach. Komunikacja z tego rodzaju aplikacjami przeprowadzana jest za pomocą sieci komputerowej, w szczególności Internetu, a także z wykorzystaniem przeglądarki internetowej. Do budowy aplikacji tego typu powinny zostać wykorzystane biblioteki i szkielet programistyczny ułatwiające pracę nad oprogramowaniem oraz jego utrzymanie, co zostało tutaj omówione. Zastosowane technologie oraz biblioteki powinny przewidywać możliwość rozbudowy aplikacji. Ponadto warto, aby konfiguracja programu umożliwiała podział zadań na większą liczbę komputerów, celem zwiększenia wydajności.

Końcowym efektem pracy są opracowane prototypy aplikacji zapewniających tworzenie formularzy i ich wypełnianie (w tym wypełnianie online), co następnie będzie prowadziło do wygenerowania wydruku wypełnionego formularza. Aplikacje będące wynikiem tej pracy zostały zaimplementowane z użyciem języka Java w wersji 5 i 6. Wykorzystano framework Struts2, biblioteki Tiles2, iText, HTTPClient oraz bazę danych PostgreSql. Dodatkowo zastosowano produkt Google Web Toolkit służący do budowy interfejsów działających w przeglądarce internetowej w oparciu o JavaScript.

SPIS TREŚCI

1. WSTĘP

Formularze są narzędziem stosowanym powszechnie w środowisku biznesowym do dokumentowania działań lub po prostu do zbierania i archiwizowania danych. Przesłanką do ich stosowania, a także ich istotną zaletą jest fakt, iż pozwalają m.in. na gromadzenie i wymianę informacji w określonym formacie. Dzięki temu umożliwiają usprawnienie różnorodnych procesów w organizacjach bądź instytucjach, począwszy od prostych administracyjno-technicznych czynności, jak przykładowo przygotowywanie umów. W rezultacie ich zastosowania należy spodziewać się zwiększenia efektywności różnorodnych procesów w przedsiębiorstwie, np. zarządzania. Formularze wykorzystywane są zarówno w środowisku biznesowym (np. przelewy bankowe, faktury, korespondencja handlowa, raporty, ankiety), jak i w czynnościach związanych z sektorem publicznym. Przykładem ich zastosowania w sektorze publicznym są formularze do wypełniania deklaracji podatkowych przez osoby fizyczne i prawne, czy też dokumenty składane do Zakładu Ubezpieczeń Społecznych (ZUS).

Wspomaganie wypełniania formularzy jest doskonałym przykładem zapotrzebowania na złożone architektonicznie oprogramowanie, które mogłoby zostać wykorzystane zarówno w małej firmie, jak i w dużym międzynarodowym koncernie. Dobrze zaprojektowana aplikacja, wykorzystująca założenia ogólnie przyjętych zasad tworzenia oprogramowania, może zostać w łatwy sposób uruchomiona w każdym przedsiębiorstwie, niezależnie od wielkości oraz skali prowadzonej przez nie działalności. Zastosowanie standardowych rozwiązań odnośnie planowania i projektowania umożliwia dogodne rozbudowywanie aplikacji, a także jej utrzymanie po uruchomieniu. Jednocześnie zachowana pozostaje wydajność oprogramowania i jego niezawodność.

Na rynku dostępne są różnorodne implementacje rozwiązania problemu wypełniania formularzy. Jednakże wiele z oferowanych pozycji oferuje ograniczoną funkcjonalność bądź jest niedopasowane do potrzeb przedsiębiorstw - na przykład pod względem kosztów zakupu danego produktu lub rozproszenia geograficznego firmy.

1.1. CEL PRACY

Celem pracy jest opracowanie zasad tworzenia oprogramowania dostępnego za pomocą Internetu i rozwiązującego dany problem biznesowy oraz wybór odpowiednich do zrealizowania tego zadania bibliotek programistycznych i technologii. W szczególności będzie to oprogramowanie wspomagające tworzenie i wypełnianie formularzy, charakteryzujące się następującymi cechami:

- Będzie zbudowane w oparciu o nowoczesne techniki programistyczne, umożliwiające rozbudowę i rozszerzenie pierwotnego zakresu możliwości przy jednoczesnym zachowaniu wysokiego poziomu wydajności i niezawodności;
- Nie będzie ograniczone do wykorzystania u konkretnego użytkownika, gdyż powinno umożliwić utworzenie własnych formularzy;
- Powinno być napisane i zaprojektowane w sposób nieograniczający użytkownika końcowego w doborze sprzętu, jak również oprogramowania, a w szczególności systemu operacyjnego;
- Technologia wykonania powinna przewidywać wzrost liczby użytkowników, a co za tym idzie, zwiększenie zapotrzebowania na wydajność i dostępność. Aby sprostać zwiększonemu zapotrzebowaniu, technologia powinna umożliwić zwielokrotnienie zasobów dostępnych dla oprogramowania np. poprzez przeniesienie części działań na inną maszynę;
- Nie będzie wymagało posiadania przez użytkownika skomplikowanych i drogich urządzeń ani kosztownego oprogramowania towarzyszącego (przykładowo konkretnego systemu operacyjnego);

- Rozwiąże problem biznesowy zdefiniowany jako: unifikacja wyglądu wydruków w rozproszonym geograficznie przedsiębiorstwie;
- Będzie wyposażone w spójny, łatwy w obsłudze (intuicyjny) interfejs użytkownika, niepowodujący trudności w obsłudze.

1.2. ROZWIĄZANIA PRZYJĘTE W PRACY

Utworzone oprogramowanie będzie składało się z kilku elementów, których zastosowanie przełoży się na osiągnięcie celu niniejszej pracy. Oprogramowanie będzie składało się z następujących projektów programistycznych:

- Edytor formularzy;
- Serwer do obsługi formularzy umożliwiający wygenerowanie wydruków, jak również zarządzanie użytkownikami i magazynowanie dostarczonych danych;
- Program eksportujący dane do serwera i pobierający gotowe wydruki.

Ze względu na różnice w budowie i zasadach działania, z części serwerowej oprogramowania będącego przedmiotem tej pracy został wyodrębniony podprojekt, który umożliwia wypełnianie formularzy online.

Oprogramowanie zostanie opracowane z wykorzystaniem języka programowania Java w wersji 5 i 6. W przypadku aplikacji uruchamianych na komputerze klienta (dwa programy z trzech będących rezultatem pracy), interfejs użytkownika będzie zbudowany w oparciu o popularną bibliotekę Awt i Swing. Część serwerowa zostanie oparta o serwer aplikacji (kontener) Apache Tomcat 6 i serwer bazy danych PostgreSql 8. Do określenia działania zostanie wykorzystany framework Struts2, natomiast do stworzenia interfejsu użytkownika - biblioteka Tiles 2 i Jsp. Część serwerowa będzie zawierała podprogram umożliwiający wypełnianie formularzy w przeglądarce internetowej, który zostanie opracowany przy użyciu narzędzia Google Web Toolkit (GWT). Uzasadnieniem dla wykorzystania GWT jest fakt, iż umożliwia on tworzenie oprogramowania działającego w przeglądarce, opartego o JavaScript, w sposób podobny do tworzenia oprogramowania okienkowego w języku Java z wykorzystaniem

biblioteki Swing. Owe podobieństwo wpływa znacząco na skrócenie czasu pracy nad aplikacją poprzez wykorzystania znanych już mechanizmów języka Java i jego biblioteki standardowej.

1.3. REZULTAT PRACY

Bezpośrednim wynikiem pracy będzie opracowanie zasad tworzenia oprogramowania dostępnego za pomocą Internetu. Jak również wybór odpowiednich technologii i bibliotek programistycznych mających ułatwiać pracę nad aplikacją i umożliwiać konfigurację systemu w sposób możliwie uniwersalny dla każdego z potencjalnych klientów. Wymagania klientów mogą różnić się od siebie i charakteryzować np. różną ilością danych, różną liczbą obsługiwanych jednocześnie użytkowników, dostępnością serwerów, już zainstalowanym oprogramowaniem, systemami operacyjnymi.

Pośrednim wynikiem pracy będzie opracowany zestaw programów wspomagający wypełnianie formularzy, cechujący się prostotą obsługi. Dzięki wykorzystaniu nowoczesnych technik programistycznych oraz skorzystania z ogólnodostępnych darmowych bibliotek i narzędzi będzie on łatwy w rozbudowie i utrzymaniu.

1.4. ORGANIZACJA PRACY

Praca rozpoczyna się od przedstawienia problemów stojących przed twórcami oprogramowania i potrzeb, na jakie odpowiada oprogramowanie stosowane w przedsiębiorstwach. Opisano wymagania, które muszą zostać spełnione, aby wdrożona aplikacja przyniosła korzyści i rozwiązała problem biznesowy zarówno w przyszłości bliższej, jak i dalszej. Opisano modelowe architektury oprogramowania, które mogą zostać wykorzystane przez twórców oprogramowania.

Rozdział trzeci prezentuje i omawia dostępne na rynku rozwiązania umożliwiające tworzenie oraz wypełnianie formularzy w formie elektronicznej, za pomocą komputera.

W rozdziale czwartym omówiono technologie wykorzystane do budowy prototypów będących wynikiem tej pracy. Przedstawiono sposób wykorzystania oraz konfiguracji

narzędzi i bibliotek. Rozdział zawiera również uargumentowanie celowości użycia i sposobu wzajemnej współpracy przedstawionych bibliotek.

Wymagania stawiane aplikacjom służącym tworzeniu i wypełnianiu formularzy prezentuje rozdział piąty. Omówiono zależności między aplikacjami z podziałem na ich funkcje oraz sposoby obsługi interfejsu.

Rozdział szósty poświęcono prezentacji wykorzystania technologii na konkretnym przykładzie aplikacji wynikowych tej pracy. Rozdział zawiera opis implementacji projektów Edytor formularzy, programu eksportującego dane do serwera, oprogramowania serwera, jak również wydzielonego projektu interfejsu użytkownika służącego do wypełniania formularzy w przeglądarce internetowej.

2. ARCHITEKTURA OPROGRAMOWANIA

2.1. WYBÓR ARCHITEKTURY OPROGRAMOWANIA

Dobór i zastosowanie odpowiedniej technologii budowy aplikacji jest rzeczą niezbędną do tego, aby produkt odniósł sukces, przy czym sukces rozumiany jest jako spełnienie oczekiwań odbiorcy, również tych przyszłych. Błędne rozpoznanie potrzeb klienta lub błędy w projektowaniu powiększają koszty tworzenia, utrzymania i rozwoju oprogramowania w późniejszych etapach pracy.

Wymagane jest, aby dobór architektury i technologii budowy aplikacji poprzedzony był rozpoznaniem potrzeb oraz wymagań przyszłego użytkownika oprogramowania. Wskazane są wspólne ustalenia twórcy i użytkownika oprogramowania odnośnie zakresu zaspokojenia tych potrzeb. Dodatkowo spotkania takie umożliwiają ocenienie nakładów pracy, a co za tym idzie zasięgu aplikacji, jak również sprzyjają określeniu wspólnej terminologii porozumiewania się. Często dopiero w czasie tego typu rozmów klient odkrywa, czego tak naprawdę potrzebuje i wówczas projekt może nabrać ostatecznego kształtu. W trosce o jakość tworzonego oprogramowania oraz udanych relacji pomiędzy zleceniodawcą a zleceniobiorcą, wspólna praca nad specyfikacją produktu jest niezbędnym etapem realizacji projektu. Następnie na podstawie wypracowanych ustaleń zawierana jest umowa wiążąca obie strony. Niespełnienie niedopowiedzianych wymagań klienta może zakończyć się problemami przy odbiorze produktu oraz zachwianiem relacji biznesowych.

Z wyżej wymienionych względów warto również prowadzić prace programistyczne w bliskim kontakcie z klientem, dostarczając co pewien czas wersje częściowo ukończonego produktu do konsultacji. Możliwie szybkie zidentyfikowanie i zrozumienie niedomagań prototypu umożliwi bowiem odpowiednio szybką reakcję. Wprowadzenie stosownych poprawek we wcześniejszej fazie projektu zdecydowanie ogranicza koszty po stronie twórców oprogramowania. Należy mieć na uwadze, iż te same zmiany wykonywane na późniejszych etapach prac będą wymagały przebudowania i wprowadzenia zmian w większej części programu, więc w rezultacie będą bardziej czaso- i kosztochłonne. Inną korzyścią płynącą z bliskiej współpracy i częstych konsultacji z klientem może być jego zadowolenie wynikające z poczucia nadzoru nad projektem. Źródłem satysfakcji dla nabywcy oprogramowania będzie także obserwacja postępu prac oraz wdrażania jego oczekiwań i wyobrażeń związanych z produktem. Bieżąca współpraca może również zaowocować tym, iż w przypadku wystąpienia zagrożeń i/lub opóźnień w realizacji projektu, klient będzie prezentował bardziej elastyczne podejście do warunków wykonania własnego zlecenia. Dzięki temu zleceniodawca może być bardziej skłonny do ograniczenia/podziału zakresu prac wymagalnych w danej jednostce czasu, bądź do wydłużenia czasu realizacji projektu. Elastyczne podejście ze strony zamawiającego pozwoli zakończyć prace i nie doprowadzi do upadku projektu.

2.2. PRZEGLĄD ISTNIEJĄCYCH ARCHITEKTUR

Wzrost zapotrzebowania na bardziej skomplikowane sposoby przetwarzanie informacji, a co za tym idzie wzrost zapotrzebowania na wydajność, niezawodność i dostępność systemów informacyjnych wspierających różnorodne procesy oraz operacje, przyczynił się do rozwoju coraz to bardziej złożonych architektur oprogramowania.

Rozwój ten został podyktowany, po pierwsze, przez wzrastającą złożoność otoczenia zewnętrznego organizacji, które jest uwzględniane przez program. O środowisku, w którym prowadzona jest działalność jednostek/organizacji, decydują czynniki takie jak: uwarunkowania międzynarodowe, polityczne, prawne, ekonomiczne czy społeczne. Oddziałują one silniej bądź słabiej na przedsiębiorstwo, ustalając ramy

jego działalności. Z tego względu mogą one sprzyjać osiągnięciu sukcesu przez firmę bądź stanowić dla niej źródło zagrożeń. Wynika stąd konieczność gromadzenia i przetwarzania danych o zewnętrznych czynnikach warunkujących działanie przedsiębiorstwa. Jednakże nie jest to łatwym zadaniem, m.in. ze względu na dużą liczbę danych, które mogą być pozyskane „z zewnątrz". Dodatkową trudnością są liczne powiązania między gromadzonymi danymi, a także konieczność poddania ich analizie/obróbce, bez której uzyskanie pożądanej informacji byłoby niemożliwe. Analogicznie istotny wpływ na wynik działalności danej organizacji mają jej uwarunkowania wewnętrzne, co do których również istnieje potrzeba zbierania danych i ich późniejszej analizy.

Biorąc pod uwagę powyższe, zidentyfikowana została potrzeba wynalezienia wydajnych sposobów podziału aplikacji na mniejsze podsystemy, nad których opracowywaniem i użytkowaniem łatwiej zapanować. Ponadto potrzeba ta wynika z dążenia do poprawy wydajności systemów informacyjnych służących opisowi skomplikowanej rzeczywistości, bądź jej wycinka.

2.2.1. Architektura jednowarstwowa

W skład architektury jednowarstwowej wchodzi najczęściej jeden program, który nie komunikuje się ze światem zewnętrznym. Program taki, aby korzystać z danych zapisanych na dysku lokalnym lub sieciowym, musi posiadać wbudowany silnik bazy danych. Pliki bazy, jeżeli są dostępne na dysku sieciowym, często muszą zostać w całości przetransferowane przez sieć do komputera, na którym mają być wykorzystane. Skutkuje to sytuacją, w której pliki bazy znajdujące się na zdalnym dysku muszą zostać zablokowane w tym czasie przed zapisem dla innych użytkowników. Po zmodyfikowaniu danych w pliku może on zostać ponownie zapisany na dysku sieciowym. Odnośnie zapewnienia spójności danych w tym modelu bazodanowym powstało wiele rozwiązań, polegających głównie na ograniczeniu do minimum okresów, gdy pliki są zablokowane.

Schemat 1 Schemat architektury jednowarstwowej

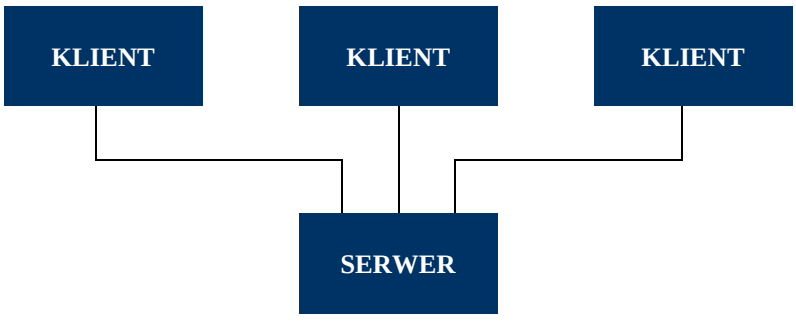
Źródło: Opracowanie własne

Programy tej architektury mogą działać często tylko na jednym komputerze lub w sieci lokalnej, która oferuje dużą przepustowość.

2.2.2. Architektura dwuwarstwowa

W architekturze dwuwarstwowej wyróżnia się program klienta i program serwera. Program klienta wykorzystuje do pracy usługi udostępniane przez serwer. Oprogramowanie może być uruchomione na jednym komputerze, współdzielonym przez programy, jak również współpracować przez sieć. W tym wypadku istnieje możliwość odciążenia programu klienta z logiki aplikacji i przeniesienia jej do programu serwera.

Dodatkową możliwością jest fakt, iż w wypadku zastosowania architektury dwuwarstwowej wiele programów typu klient może korzystać z jednego serwera. Co więcej, program serwer może być rozbudowywany lub modyfikowany niezależnie od programów klienta. Zmiana działania serwera z zachowaniem jego zewnętrznego interfejsu nie powoduje potrzeby zmian programów klienckich.



Schemat 2 Schemat architektury dwuwarstwowej

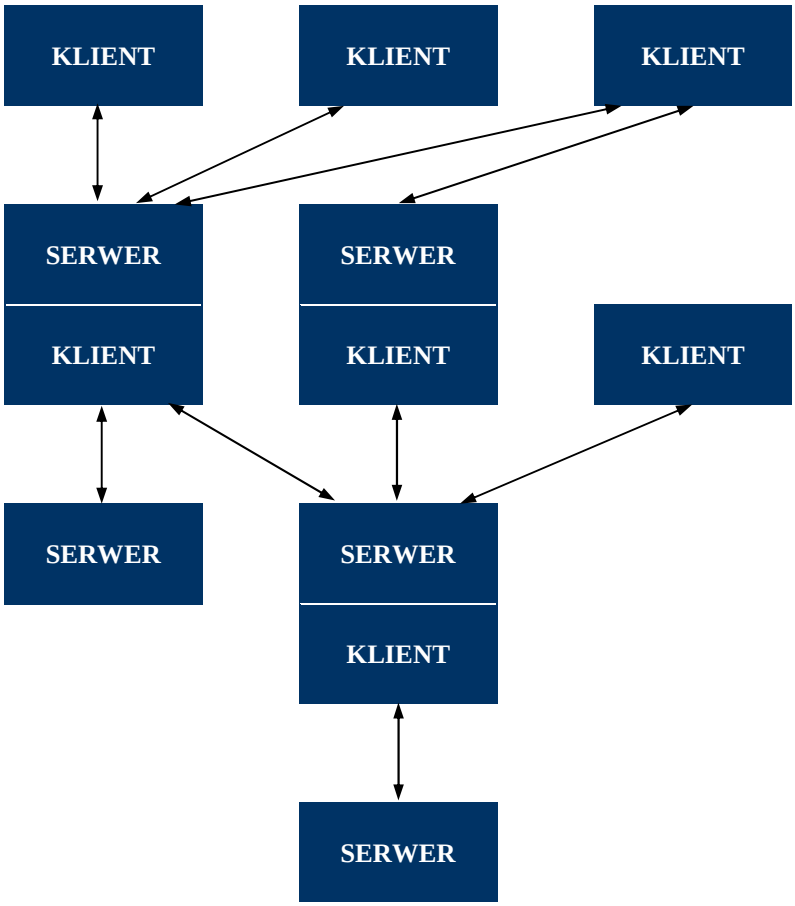
Źródło: Opracowanie własne

2.2.3. Architektura wielowarstwowa

O architekturze wielowarstwowej, rozumianej w tym wypadku jako architekturze trój- i więcej warstwowej, możemy mówić w sytuacji, gdy serwer występujący w architekturze dwuwarstwowej staje się klientem innych serwerów.

Serwer wykorzystuje usługi dostarczane przez inne serwery. Takie rozwiązanie ułatwia budowę aplikacji, których zakres obejmowanego środowiska zewnętrznego mającego wpływ na działalność przedsiębiorstwa jest bardzo duży. Rozbudowane tematycznie systemy komputerowe przetwarzają wielkie ilości danych. Zastosowanie tej architektury daje możliwość podzielenia problemu na mniejsze kawałki i rozmieszczenie ich na różnych maszynach. Skutkuje to zwiększeniem mocy obliczeniowej, możliwością jednoczesnej obsługi większej liczby użytkowników, jak również zwiększeniem prędkości odpowiedzi systemu.

Przykładowe rozwiązanie z zastosowaniem architektury wielowarstwowej zostało zaprezentowane na poniższym schemacie *(Schemat 3*).



Schemat 3 Schemat architektury wielowarstwowej

Źródło: Opracowanie własne

3. STAN SZTUKI

Rynek oprogramowania nakierowanego na ułatwianie pracy i zmniejszanie kosztów działalności firm jest mocno rozbudowany. Dostępne są rozwiązania wspomagające większość działań wykonywanych w ramach prowadzenia firmy. Także segment rynku odpowiedzialny za dostarczanie oprogramowania do wydruków i wypełnianie formularzy nie jest niszowym. Rynek obfituje w różnorodne rozwiązania - od najprostszych do bardzo rozbudowanych. Dostępne programy różnią się między sobą funkcjonalnością i zastosowanymi technologiami. Najprostsze umożliwiają wyłącznie wypełnianie formularzy, podczas gdy najbardziej rozbudowane oferują możliwości przygotowywania formularzy, wypełniania ich za pomocą różnych technologii, zbierania danych w bazach danych, przygotowywanie raportów statystycznych itd.

3.1. ADOBE ACROBAT READER, PROGRAMY DO WYPEŁNIANIA ZEZNAŃ PODATKOWYCH

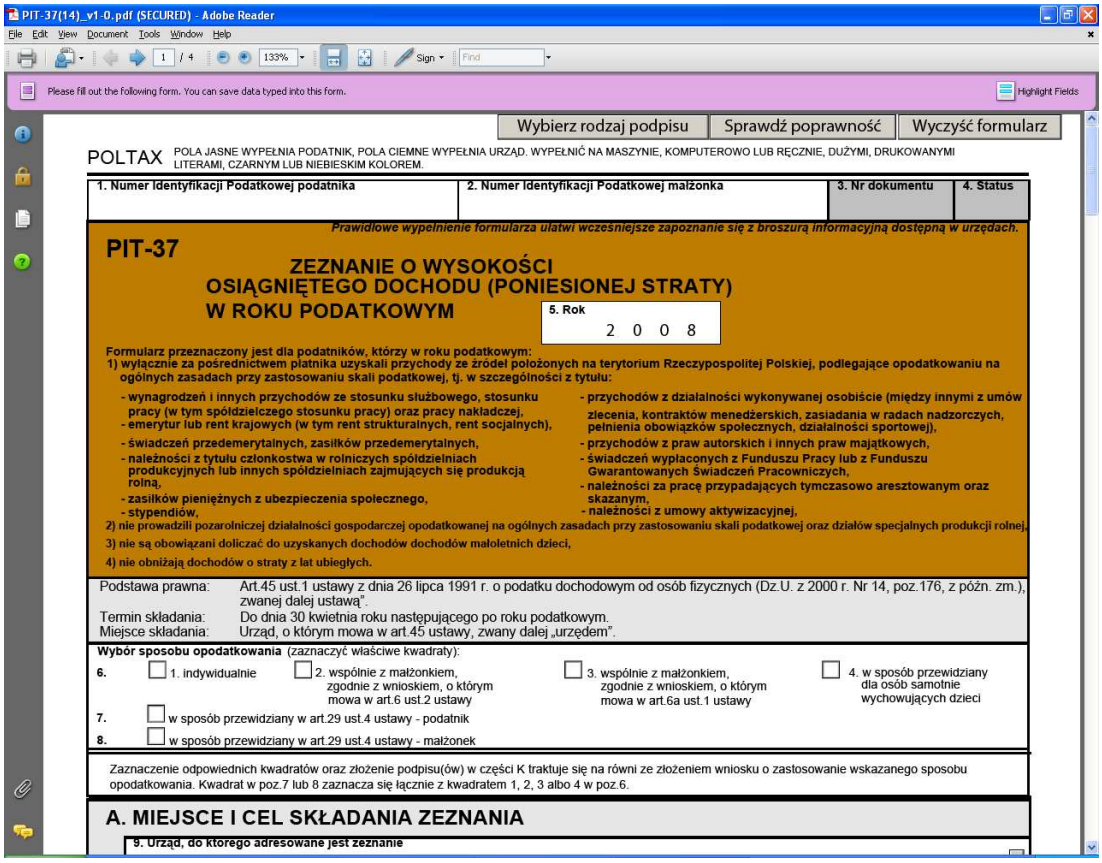
Adobe Acrobat Reader jest programem ogólnie znanym, dostępnym za darmo, z reguły używanym do wyświetlania i drukowania dokumentów w formacie PDF. Jedną z jego funkcji jest możliwość wyświetlenia dokumentu/formularza PDF przygotowanego z wykorzystaniem obiektów będących polami edycyjnymi, w które użytkownik może wpisać potrzebne dane. W ten sposób wypełniony dokument można wydrukować. W roku 2009 Ministerstwo Finansów RP umożliwiło pobranie ze strony http://www.e-deklaracje.gov.pl/ wielu gotowych formularzy PDF służących osobom fizycznym i podmiotom gospodarczym do wymiany informacji z urzędami skarbowymi.

Taki sam sposób podejścia do problemu oferują programy dostępne od wielu lat w sieci Internet służące do wypełniania rocznych zeznań podatkowych dla osób fizycznych. Często są one rozprowadzane za pomocą popularnych serwisów informacyjnych np. www.gazeta.pl, www.onet.pl. Programy posiadają zaimplementowane mechanizmy wypełniania formularzy, które są wbudowane w program. Najczęściej umożliwiają również obliczenie sum, różnic, procentów, co dzieje się automatycznie po wprowadzeniu danych w odpowiednie miejsca. Obliczone dane pojawiają się w zdefiniowanych przez autora programu komórkach formularza.

Rozwiązania tego rodzaju, choć proste w obsłudze, oferują ułatwienie pracy jedynie osobom zainteresowanym wypełnieniem formularza. Instytucja lub osoba otrzymująca wypełniony i wydrukowany formularz musi wykonać pełny zakres prac identyczny z tym, który jest wykonywany po otrzymaniu formularza wypełnionego ręcznie. Najczęściej jest to sprawdzenie oraz przepisanie danych do systemu komputerowego zbierającego i przetwarzającego te dane. W tym względzie nastąpiła zmiana w rozliczeniach z Ministerstwem Finansów RP (MF) za rok 2008 i 2009. Kwestia ta zostanie przybliżona w dalszej części niniejszego rozdziału.

Kolejną funkcją, której brak może odczuwać użytkownik w tego rodzaju rozwiązaniu, jest możliwość utworzenia swojego własnego formularza. Pliki wykorzystywane przez program Acrobat Reader muszą być przygotowane przez doświadczoną osobę w specjalnym, często niedarmowym oprogramowaniu (np. Adobe Acrobat). Programy ułatwiające wypełnianie rozliczenia podatkowego ograniczają się tylko do konkretnych formularzy wbudowanych w program. Często nie można rozszerzyć listy wbudowanych formularzy, konieczna jest wtedy instalacja osobnego programu, który będzie je zawierał.

Na poniższym zdjęciu (*Rysunek 1*) przedstawiono program Adobe Acrobat Reader z tworzonym formularzem PIT-37.



Rysunek 1 Adobe Acrobat Reader

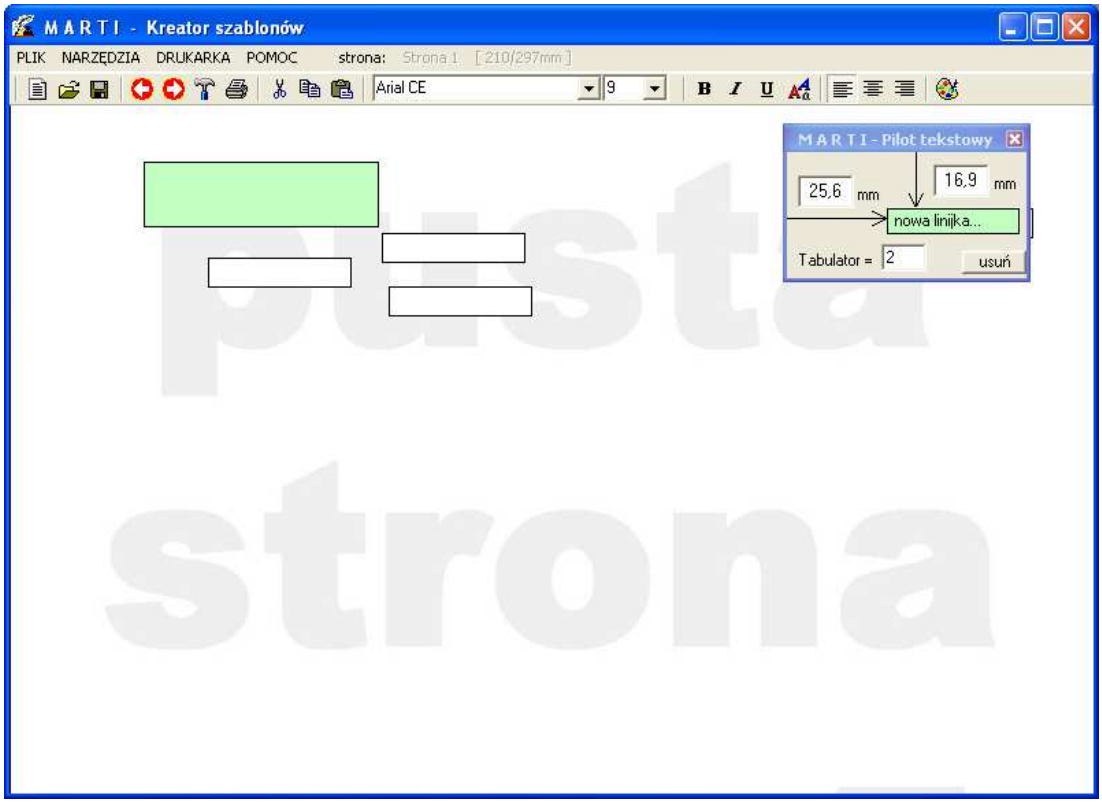
Źródło: Opracowanie własne

3.2. AUTOMINTEL - "MARTI – FORMULARZE"

Oprogramowanie opracowane przez firmę Automintel oferuje przyjemne i przejrzyste środowisko pracy. Do podstawowych funkcji należy w tym programie tworzenie formularzy, jak również ich późniejsze wypełnianie. Tworzenie formularzy polega na zeskanowaniu pustego papierowego formularza, a następnie wstawieniu go jako tło formularza w programie. Na podstawie tła można określić, w jakich miejscach powinny znajdować się pola z danymi, zaznaczenie których umożliwia program. Tak przygotowany formularz wypełniony danymi może zostać uzupełniony i wydrukowany w całości, bądź na oryginalnym papierowym formularzu mogą zostać wydrukowane wyłączenie dane, które trafią w odpowiadające im miejsca.

Program "Marti – Formularze" jest doskonałym narzędziem do przygotowywania i wypełniania formularzy, jednakże działanie programu wspiera wyłącznie stronę przygotowującą i wypełniającą formularz. Osoba otrzymująca formularz musi wykonać ręczną pracę, aby raz już przecież wprowadzone dane ponownie wprowadzić do systemu komputerowego.

Wygląd aplikacji pokazano na poniższym zdjęciu (*Rysunek 2*).



Rysunek 2 Automintel Marti Formularze

Źródło: Opracowanie własne

3.3. E-DEKLARACJE

Ministerstwo Finansów RP umożliwiło elektroniczne składanie rozliczeń podatkowych za rok 2007 i 2008 od osób fizycznych. Jednakże przy rozliczeniu za rok 2007, koniecznym potwierdzeniem tożsamości składającego deklarację był elektroniczny autoryzowany (bezpieczny) podpis cyfrowy. To rozwiązanie niestety

nie przyczyniło się do popularności tej formy rozliczeń, gdyż wejście w posiadanie bezpiecznego podpisu wiąże się z kosztem rzędu kilkuset złotych. W rozliczeniach za rok 2008 umożliwiono autoryzację na podstawie danych archiwalnych (suma dochodu z zeszłego roku) oraz tych zawartych w standardowym formularzu PIT-37 np. NIP, imię, nazwisko, PESEL, adres. Rezygnacja z wykorzystania bezpiecznego podpisu poskutkowała dużym zainteresowaniem podatników elektroniczną formą przekazywania dokumentów informujących o dochodach za roku 2008.

Oprogramowanie zostało utworzone z wykorzystaniem popularnego oprogramowania Acrobat. Działanie aplikacji jest powiązane z technologią PDF (Acrobat Reader) dostarczaną przez firmę Adobe. Do zapewnienia łączności za pomocą Internetu i możliwości złożenia zeznania podatkowego w sposób elektroniczny wykorzystano Adobe AIR (Adobe Integrated Runtime). Warto zaznaczyć, że PDF jest standardem ISO 32000.

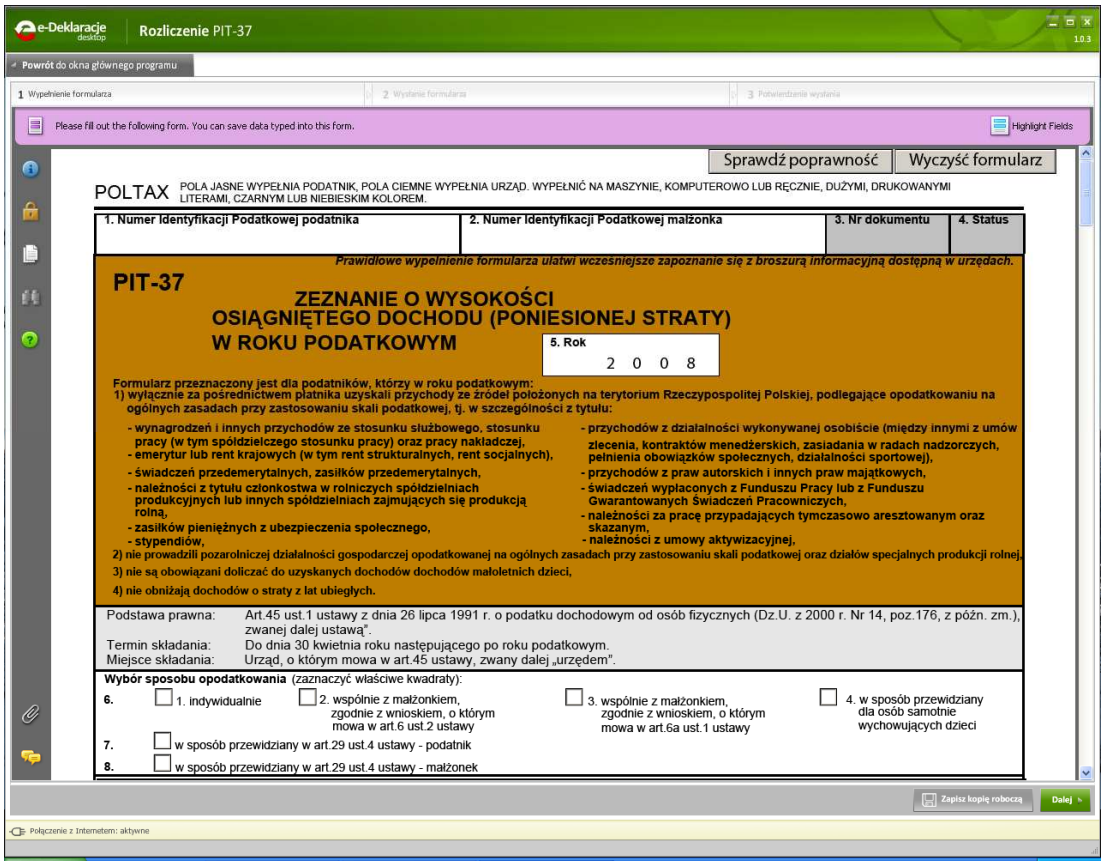
Do wypełnienia i wysłania formularza potrzebne są jedynie darmowe aplikacje Acrobat Reader i Adobe Flash (zainstalowane obecnie na ponad 90% komputerów osobistych - za tech.wp.pl). Aplikacja napisana w Adobe AIR może pracować w trzech środowiskach klienckich (Windows, MacOS, Linux).
Aplikacja oferuje ergonomiczny interfejs, jest łatwa w obsłudze, a wszystkie opcje programu są widoczne i łatwo dostępne w odpowiednich momentach. Menu aplikacji pokazano na rysunku poniżej (*Rysunek 3*).

Rysunek 3 Menu programu Ministerstwa Finansów RP e-Deklaracje

Źródło: Opracowanie własne

Po wprowadzeniu danych do rozliczenia PIT-37 sprawdzana jest poprawność (wybranych) danych. W przypadku stwierdzenia bezbłędności danych następuje zapisanie ich na dysku twardym komputera. Istnieje również możliwość wydrukowania wypełnionego formularza. Następnym krokiem jest wysłanie danych za pomocą Internetu do Urzędu Skarbowego (Ministerstwa Finansów). Po poprawnym przesłaniu, po stronie MF następuje sprawdzenie autentyczności. Gdy odebrane dane są poprawne, a autentyczność na podstawie informacji zawartych w formularzu potwierdzona, odsyłane jest potwierdzenie odebrania dokumentu, które można wydrukować. Urzędowe potwierdzenie odbioru (UPO) jest urzędowym dokumentem równoznacznym z dostarczeniem papierowego rozliczenia podatkowego do Urzędu Skarbowego.

Aplikację w stanie umożliwiającym wypełnianie zeznania podatkowego przedstawiono na poniższym rysunku (*Rysunek 4*).



Rysunek 4 e-Deklaracje PIT-37 - Ministerstwo Finansów

Źródło: Opracowanie własne

Rozwiązanie proponowane przez MF jest zdecydowanym postępem w kwestii komunikacji z podatnikiem. Wykorzystanie standardów oraz darmowego oprogramowania, firmowanego ponadto przez znaną firmę z ugruntowaną pozycją na rynku i wieloletnią praktyką, stwarza przesłanki do powodzenia przedsięwzięcia w dłuższym okresie czasu. Można przypuszczać, iż oprogramowanie w znaczącej części przejmie pracę Urzędów Skarbowych w przyjmowaniu rozliczeń podatkowych, co powinno skutkować mniejszą liczbą pomyłek oraz redukcją kosztów pobierania podatków. Należy spodziewać się, że oprogramowanie to, po raz pierwszy w Polsce na szeroką skalę, spowoduje znaczące ograniczenie wymiany informacji z Urzędami Skarbowymi w formie papierowej. Oprogramowaniem, które spełnia podobne funkcje, jednak nie przeznaczonym do użytkowania przez osoby

fizyczne, jest program “Płatnik” firmy Prokom, służący do elektronicznej wymiany informacji z Zakładem Ubezpieczeń Społecznych.

3.4. MICROSOFT

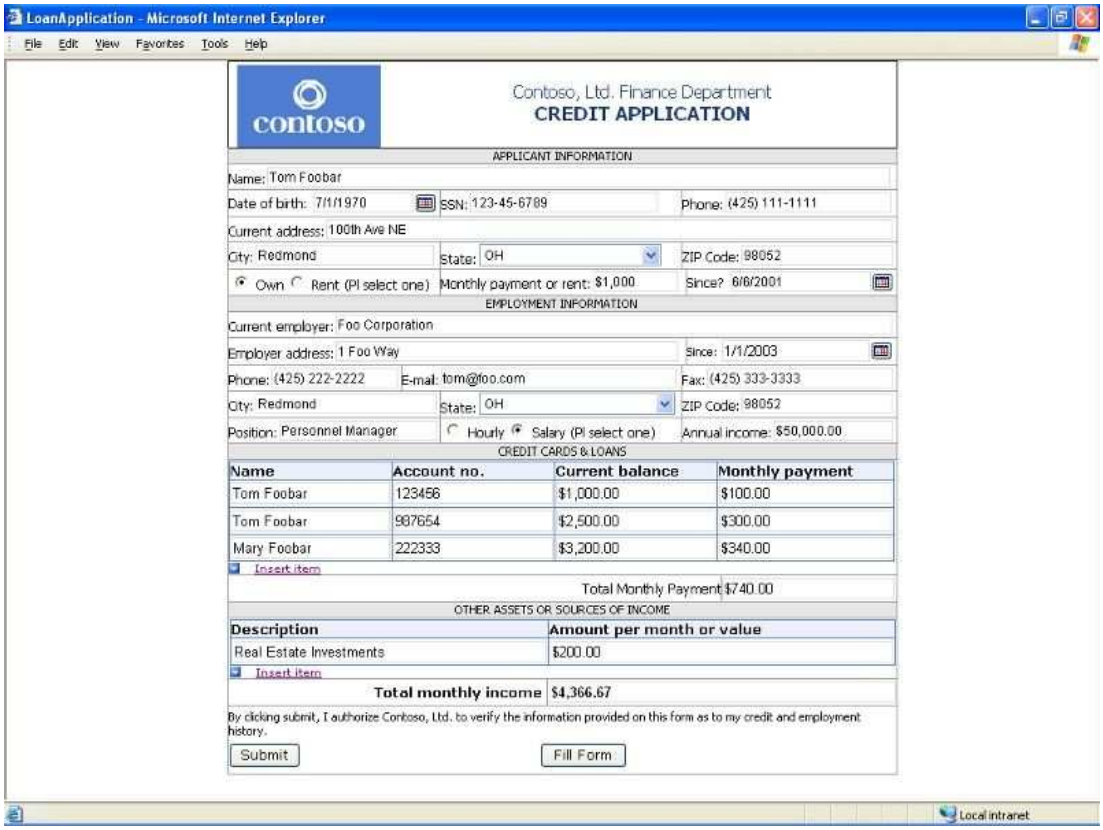
3.4.1. Microsoft Office InfoPath

MS InfoPath jest programem wchodzącym w skład pakietu Microsoft Office. Oprogramowanie to opracowano z myślą o tworzeniu i wypełnianiu formularzy. Tworzone w nim formularze mogą posiadać pola edycyjne, tabele, pola combo, itp. Jeżeli pole może być wypełnione jedynie na kilka sposobów, walidacja danych odbywa się za pomocą listy wprowadzonych wartości, bazy danych lub usługi sieciowej udostępniającej odpowiednie informacje. Ważnym elementem tego oprogramowania jest możliwość tworzenia formularzy wykonujących niestandardowe czynności dodatkowe poprzez udostępnienie możliwości połączenia formularza z kodem programu napisanego w Jscript, Visual Basic Script lub C#. Formularz można również opracowywać jako projekt w Microsoft Visual Studio, gdzie z zastosowaniem kodu zarządzanego możliwe jest osiągnięcie niestandardowego zachowania formularza.

Dane aplikacji InfoPath przechowywane są w postaci plików w formacie XML. Ta właściwość pozwala na łatwą wymianę informacji między systemami ze sobą niepowiązanymi.

3.4.2. Microsoft Windows® SharePoint™ Server

Wykorzystanie programu Microsoft SharePoint umożliwia opublikowanie formularzy w sieci komputerowej jako strony www. W takim przypadku wypełnianie formularzy odbywa się w przeglądarce, czyli inaczej niż standardowo w programie InfoPath. Wygląd formularza w przeglądarce internetowej przedstawia kolejne zdjęcie (*Rysunek 5*).



Rysunek 5 Formularz programu InfoPath w przeglądarce internetowej

Źródło: http://blogs.msdn.com/tudort/

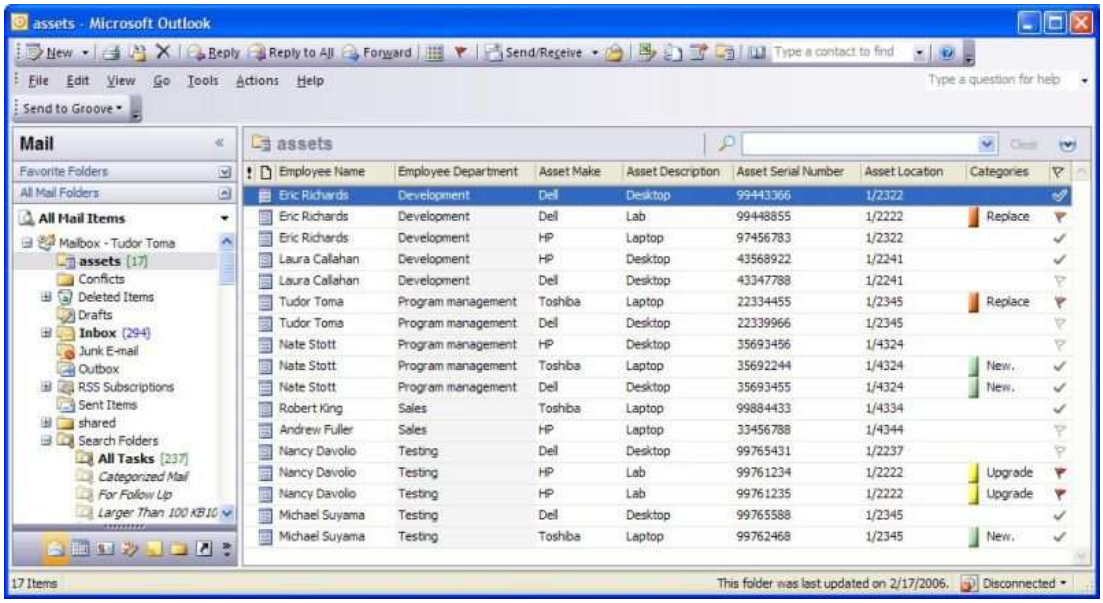
3.4.3. Microsoft Office Outlook

Przekazanie formularza do wypełnienia za pomocą programu Microsoft Outlook w postaci wiadomości e-mail umożliwia wypełnianie go offline oraz odesłanie danych w czasie, gdy komputer znajdzie się w zasięgu sieci.

Wyniki współpracy InfoPath i Outlook przedstawiono na zdjęciu na kolejnej stronie pracy (*Rysunek 6*).

Dane, które zostaną wprowadzone do formularzy i będą dostarczone do programu InfoPath, mogą zostać zapisane w postaci XML. Dane te można również zapisać w bazie danych Access lub Microsoft SQLServer. Standardowa funkcjonalność pakietu Microsoft Office zezwala na wykorzystanie tak dostarczonych danych do

tworzenia różnorodnych analiz i statystyk, a co za tym idzie, również prezentacji, wykresów i tabel.

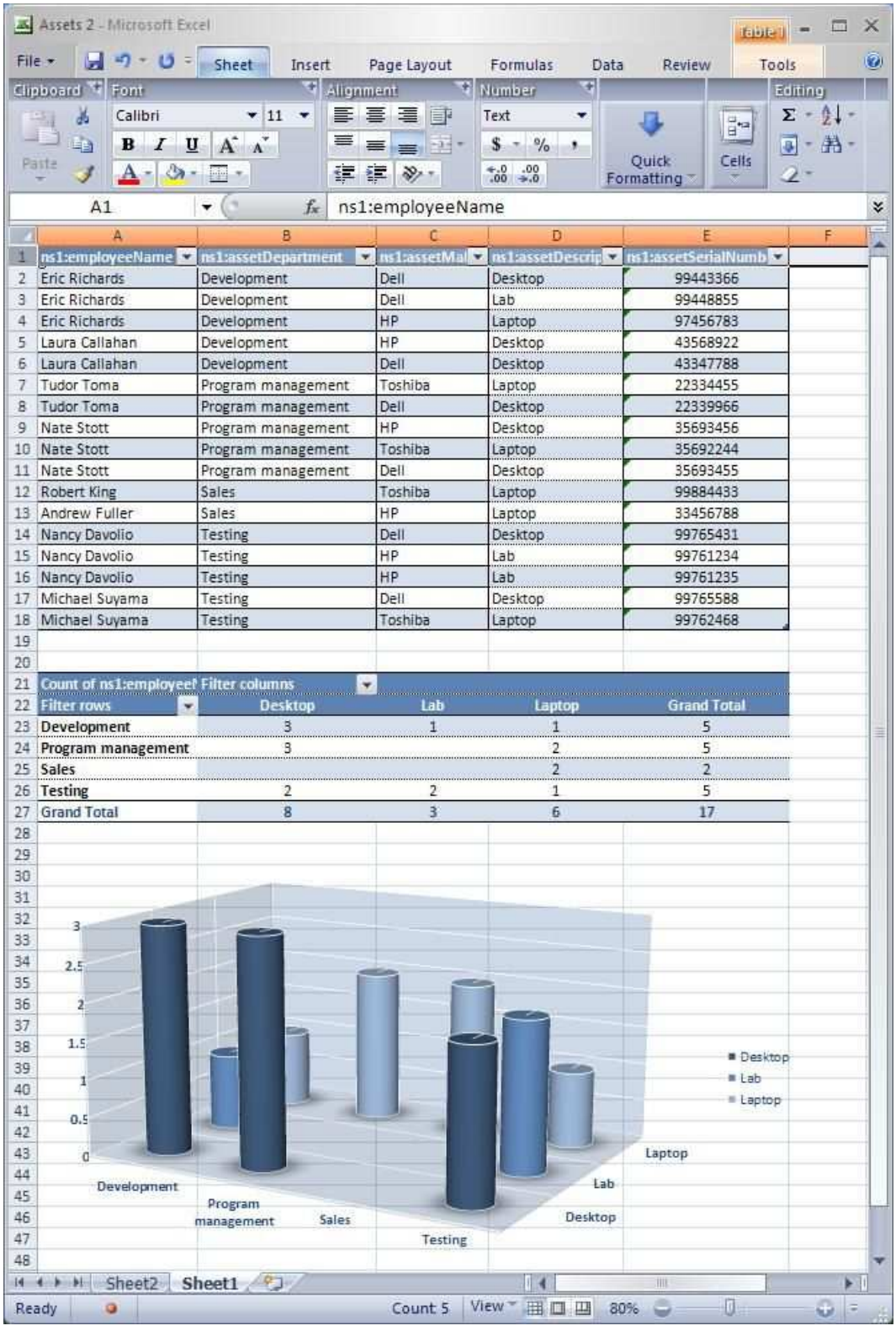


Rysunek 6 Współpraca programu Microsoft Outlook i InfoPath

Źródło: http://blogs.msdn.com/tudort/

Wyniki analizy zebranych danych przedstawia kolejny rysunek *(Rysunek 7*) na kolejnej stronie niniejszej pracy.

Wadą rozwiązania firmy Microsoft jest niedostosowanie pakietu Office do pracy z systemami operacyjnymi spoza rodziny Windows. Często przecież spotykane są serwery lub zastane w firmach bazy danych działające w środowisku Linux/Unix. Możliwość pracy tego oprogramowania wyłącznie w środowisku Windows w pewien sposób ogranicza liczbę klientów zainteresowanych tym rozwiązaniem.



Rysunek 7 Analiza danych pochodzących z formularzy InfoPath

Źródło: http://blogs.msdn.com/tudort/

3.5. PODSUMOWANIE

Podsumowując, na rynku dostępne są rozwiązania oferujące odpowiednią funkcjonalność nawet dla bardzo wymagających klientów. Niestety są wśród nich także takie pozycje, które narzucają wykorzystanie systemu operacyjnego Microsoft Windows. Można stwierdzić również, że zauważalny jest ruch w kierunku rozwiązań dostępnych dla użytkowników pracujących pod różnymi systemami operacyjnymi. Rozwiązania firm wiodących, takich jak Microsoft czy Adobe, umożliwiają uzupełnienie formularzy o kawałki programu stworzone samodzielnie, co zwiększa uniwersalność rozwiązania.

4. OPIS NARZĘDZI ZASTOSOWANYCH W PRACY

4.1. JAVA

Java jest językiem obiektowym skonstruowanym na podstawie składni C i C++, jednakże w jego budowie zaniechano rozwoju konstrukcji niskopoziomowych. Java została opracowana przez Sun Microsystems w 1995 roku. Jest językiem programowania, który umożliwia pisanie programów niezdeterminowanych architekturą i systemem operacyjnym maszyny wykonawczej. Kompilacja programów Javy następuje dwuetapowo. Pierwszy etap odbywa się na komputerze programisty, a polega na produkcji częściowo skompilowanego kodu - z ang. *byte code*. Druga część kompilacji jest wykonywana przez Maszynę Wirtualną Javy (z ang. *Java Virtual Machine* – JVM) podczas uruchomienia programu. To rozwiązanie umożliwia uruchamianie oprogramowania na różnych systemach operacyjnych, gdyż dla każdego systemu operacyjnego jest dostarczana inna JVM wykonująca kompilację końcową. Dzięki wykorzystaniu JVM, która jest pośrednikiem między oprogramowaniem a systemem operacyjnym, zyskuje się większy poziom bezpieczeństwa systemu. JVM uruchamia program z ograniczonymi prawami do ingerencji w system operacyjny. W przypadku niestabilności programu działającego wewnątrz JVM, system jest zabezpieczony przed propagacją błędu. JVM jest również wyposażona w mechanizm zarządzania pamięcią przydzieloną oprogramowaniu zwaną z ang. *Garbage Collecion* (GC). Mechanizm ten ma za zadanie odciążyć programistę z potrzeby zwalniania zaalokowanych zasobów. Co pewien interwał czasu GC sprawdza, czy do zaalokowanej komórki pamięci są

w programie przechowywane referencje; jeżeli ich nie ma, taka komórka jest zwracana do puli dostępnej pamięci.

Platforma programistyczna Java 2 Standard Edition (J2SE) w wersji 5 i 6 została wykorzystana do opracowania dwóch z trzech projektów wchodzących w skład prototypu. J2SE udostępnia bibliotekę Swing do tworzenia aplikacji okienkowych i została ona wykorzystana w Edytorze Formularzy i Programie eksportującym dane.

4.2. BIBLIOTEKA SWING

Biblioteka Swing zawiera klasy odpowiedzialne za tworzenie interfejsu użytkownika w aplikacjach okienkowych. Za jej pomocą w szybki sposób można zbudować skomplikowany interfejs składający się z okien, przycisków, pól edycyjnych, paneli, suwaków. Aplikacja napisana z wykorzystaniem Swing będzie działała na każdym komputerze z dostępnym Java Runtime Environmnt (JRE). Rozmieszczenie elementów w oknie jest kontrolowane przez menedżera wyglądu (*z ang. Layout Manager*), jednego z dostępnych w bibliotece. Interfejs aplikacji jest zbudowany z głównego okna, które jest kontenerem dla mniejszych komponentów. Komponenty składające się na interfejs są powiadamiane o zaistniałych zdarzeniach (np. najechanie wskaźnika myszki, przyciśnięcie klawisza) poprzez mechanizm Listenerów. Mechanizm Listener opiera się na obiektach implementujących konkretny interfejs Listener. Obiekty te umożliwiają odpowiednie reagowanie na zdarzenia pochodzące z interfejsu obsługiwanego przez użytkownika.

4.3. JAVA ENTERPRISE EDITION

Serwerowa część prototypu została oparta również o język Java w wersji 5. W tej części pracy wykorzystana została specyfikacja Java Enterprise Edition (JEE) (patrz [1]). JEE jest zbiorem zasad, specyfikacji, którymi powinny kierować się osoby tworzące oprogramowanie klasy „korporacyjnej”. Implementacje tych specyfikacji są często darmowe, jak również są udostępniane jako oprogramowanie o otwartym kodzie źródłowym, często rozwijane przez społeczność internetową (z ang. *Open Source*). Aplikacje klasy Enterprise są uruchamiane na serwerach w tak zwanych

kontenerach/serwerach aplikacji. Kontenery aplikacji są to programy dostarczające różnorodnych usług oprogramowaniu w nich działającemu. W tej pracy został wykorzystany kontener aplikacji Apache Tomcat w wersji 6. Apache Tomcat jest to niepełny serwer aplikacyjny JEE, który jednak spełnia wymagania stawiane przez ten prototyp. Opracowywany program nie korzysta ze wszystkich możliwości wynikających ze specyfikacji JEE, przez co jego zapotrzebowanie na zasoby systemowe udostępniane przez kontener jest mniejsze. Również z tego powodu czas uruchamiania lub restartowania tego kontenera jest krótszy. Wyjaśniając, niepełnym serwerem aplikacyjnym JEE jest taki kontener, który nie udostępnia wszystkich usług określonych w specyfikacji JEE.

Apache Tomcat przez długi czas był częścią projektu Apache Jakarta, lecz od pewnego momentu jest rozwijany przez organizację Apache jako osobny projekt. Tomcat jest elementem wchodzącym w skład serwera HTTP Apache (patrz [2]), jak również serwera Apache Geronimo. Z powodu swojego małego rozmiaru i niewielkich wymagań jest dostarczany jako serwer deweloperski z konkretnymi wydaniami środowiska programistycznego dla Javy, Eclipse.

4.4. STRUTS2

Szkielet programistyczny (z ang. *framework*) Struts2 został wykorzystany w tej pracy (więcej informacji można znaleźć w [3]) do zapewnienia poprawnego sterowania interakcją z użytkownikiem i budową aplikacji. Jest to produkt dość młody i dopiero niedawno udostępniony do szerszego wykorzystania przez programistów. Jest rozprowadzany jako produkt darmowy na licencji Apache 2.0.

Struts2 powstał z połączenia najlepszych cech frameworka WebWork i pierwszej wersji Struts, która przez wiele lat zyskała sobie zadowolenie twórców oprogramowania. Charakteryzuje się scentralizowaną konfiguracją sterowania zarówno interakcją z użytkownikiem, jak i działaniami, które oprogramowanie powinno wykonać przed lub po tej interakcji (mechanizm interceptorów). Zapewnia również łatwą w zastosowaniu konfigurację poszczególnych punktów wejścia i wyjścia z konkretnego miejsca w programie.

4.4.1. Konfiguracja i zasady działania Struts2

Konfiguracja Struts2 jest stosunkowo rozbudowaną strukturą, którą można podzielić ze względu na pełnione funkcje na kilka oddzielnych plików, a następnie włączyć do głównego pliku konfiguracyjnego za pomocą komend ‘include’. Służy to ułatwieniu pracy programisty. Warto zaznaczyć, iż zespół programistyczny zaangażowany w tworzenie Struts2 zaimplementował możliwość dziedziczenia ustawień konfiguracyjnych. W przypadku tworzonego w ramach tej pracy prototypu dziedziczenie odbywa się z dostarczonej razem z biblioteką konfiguracji domyślnej o nazwie struts-default. Zastosowanie dziedziczenia z wykorzystaniem domyślnej konfiguracji zdejmuje z programisty potrzebę zdefiniowania wszystkich wartości domyślnych niezbędnych do uruchomienia programu. Po przeprowadzeniu dziedziczenia można oczywiście przesłonić niektóre elementy konfiguracji domyślnej i nadać im nowe, odpowiednie dla projektu wartości.

Plik konfiguracyjny powinien nazywać się ‘struts.xml’ i być widoczny dla serwletu konfiguracyjnego. Plik składa się z definicji:

- Akcji;
- Interceptorów;
- Stosu intereceptorów;
- Typów wyników akcji.

4.4.2. Akcje

Podstawową jednostką logiczną i ustanawianą w konfiguracji Struts2 jest akcja. Akcja zostaje zdefiniowana poprzez tag <action>. Najczęściej wykorzystywanymi parametrami akcji są:

- Nazwa akcji – obowiązkowo;
- Klasa java/pojo odpowiedzialna za działanie i udostępnienie danych – opcjonalnie;
- Określenie metody powyższej klasy, która powinna zostać wykonana w momencie wywołania danej akcji – opcjonalnie.

Takie wymagania konfiguracyjne umożliwiają przygotowanie klas, które będą wykorzystywane przez kilka akcji, różniąc się jedynie przypisanymi do nich wykonywanymi metodami. Definicja akcji umożliwia również zastosowanie znaku wieloznacznego w każdej z sekcji tej definicji, co znacznie ułatwia tworzenie podobnych akcji typu np. *create, read, update i delete* (CRUD). Przykład konfiguracji akcji z wykorzystaniem znaku wieloznacznego wygląda następująco:

```
<action name="User_*" class="web.UserAction" method="{1}">
</action>
```

Gdy korzystamy ze sposobu konfiguracji z użyciem znaków wieloznaczności, dobrą praktyką jest zdefiniowanie akcji przechwytującej wywołanie akcji o nazwie niepasującej do żadnego ze zdefiniowanych wzorców, a następnie przekierowującej np. na stronę błędu.

Klasa akcji jest zwykłą klasą Java, która może zawierać dowolne pola i metody. Klasa akcji powinna być klasą publiczną widoczną dla mechanizmu tworzenia obiektów akcji Struts2. Jedynym ograniczeniem stawianym klasom akcji jest przymus, aby metody wykorzystane w konfiguracji były publiczne i zwracały wynik w postaci String. Wynik metody akcji jest wykorzystywany przez Struts2 do wygenerowania właściwej w danym momencie odpowiedzi dla użytkownika.

4.4.3. Definicja akcji

Definicja akcji określa, pod jakim adresem wywoływanym przez użytkownika w przeglądarce powinna pojawić się pożądana strona, a także jakie są jej wyniki. Definicja określa nazwę akcji, klasę, która zawiera kod programu i definiuje zachowanie strony, metodę tej klasy odpowiedzialną za dane na stronie, jak również możliwe sposoby wyjścia z akcji, czyli jej wyniki. Klasa akcji jest zwykłą klasą Javy Plain Old Java Object (*POJO*), która posiada pola i metody. Metody są odpowiedzialne za odpowiednie działania oraz manipulację polami, z których korzysta strona JSP, za pomocą getterów i setterów do wygenerowania HTML.

Definicja akcji wygląda następująco:

```
<action name="Login" class="web.Login">
```

```
  <result name="input">/site/tiles/Logowanie_t.jsp</result>
  <result name="success" type="redirect-action">Menu</result>
</action>
```

natomiast przykładowa klasa może wyglądać następująco:

```
public class Login{
    public String execute() throws Exception {
        return „success”;
    }
    private String username;
    private String password;
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Jeżeli klasa akcji implementuje interfejsy dostarczane razem z interceptorami, przed wywołaniem jakiejkolwiek metody akcji zostanie obsłużona przez interceptor. W szczególności klasa akcji zostanie uzupełniona danymi. Implementacja interfejsu przynależnego do konkretnego interceptora gwarantuje, iż klasa akcji posiada właściwe metody wykorzystywane przez ten interceptor.

Metody akcji wykorzystane w jej definicji muszą zwracać wynik w postaci String. Jest on informacją dla Struts2, który ze zdefiniowanych wyników powinien zostać wykonany. Jeżeli definicja akcji nie wskazuje metody akcji, która powinna zostać wykonana, domyślnie zostanie wywołana metoda ‘execute’.

4.4.4. Wynik akcji

Wynik akcji jest rozpoznawany po nazwie zwróconej jako wynik wywołania metody akcji. Biblioteka Struts2 udostępnia kilka zdefiniowanych typów wyników i lista ta nie jest zamknięta. Programista może rozszerzyć listę o typ wyniku stworzony przez siebie.

Parametrami są:

- Nazwa wyniku;
- Typ wyniku;
- Wartość będąca zgodna z typem wyniku.

Przykładowa konfiguracja własnego typu wyniku akcji wygląda następująco:

```
<result-types>
         <result-type name="tiles"
         class="org.apache.struts2.views.tiles.TilesResult"
         default="true"/>
</result-types>
```

W tej przykładowej konfiguracji wynik typu ‘tiles’ będzie obsługiwany przez klasę org.apache.struts2.views.tiles.TilesResult, która wygeneruje odpowiednią odpowiedź pobierając niezbędne informacje z konfiguracji dla biblioteki Tiles2.

Powyższa konfiguracja typu wyniku akcji jest elementem podłączenia biblioteki Tiles2, co daje w efekcie możliwość bezpośredniego wykorzystania jej możliwości w połączeniu ze Struts2. Wynik akcji tego typu zostanie odszukany w konfiguracji Tiles2 i przekształcony w HTML, który zostanie odesłany użytkownikowi.

Jeżeli w definicji akcji nie zaznaczono inaczej, domyślnym typem wyniku akcji jest ‘dispatcher’, który oznacza, że parametrem powinien być adres pliku Java Server Pages (JSP).

4.4.5. Interceptory i stos interceptorów

Interceptory są to klasy wywoływane w momencie, kiedy obiekt akcji został już utworzony, ale żadna z jego metod nie była jeszcze wywoływana, a także po zakończeniu wykonywania metody akcji. Interceptory umożliwiają przykładowo przypisanie wartości polom obiektu akcji, których nazwa jest identyczna z polami parametrów wywołania GET lub POST, a także sprawdzenie zalogowania użytkownika i odpowiednie przekierowanie w przypadku niezalogowania.

Stos interceptorów jest to zbiór wywoływanych kolejno interceptorów. Każda z akcji może posiadać swój stos interceptorów, jak również mogą one korzystać ze

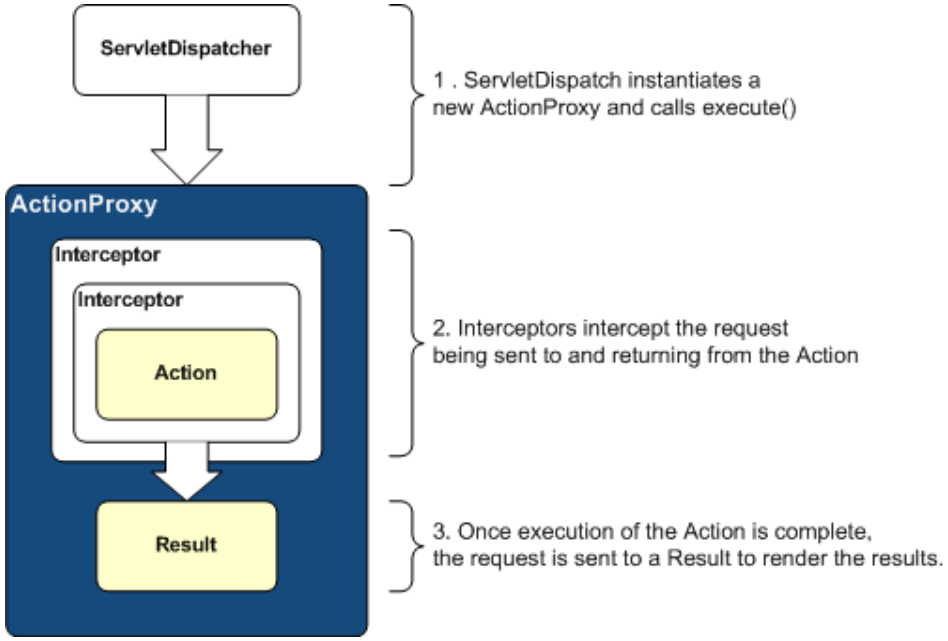
wspólnego, raz zdefiniowanego stosu. Przykładowa konfiguracja interceptora i stosu interceptorów wygląda następująco:

```
<interceptors>
  <interceptor name="login"
       class="pl.toskl.webapp.struts2.LoginInterceptor" />
  <interceptor-stack name="loginStack">
    <interceptor-ref name="servlet-config" />
    <interceptor-ref name="exception"/>
    <interceptor-ref name="alias"/>
    <interceptor-ref name="params"/>
    <interceptor-ref name="prepare" />
    (…)
    <interceptor-ref name="login" />
  </interceptor-stack>
</interceptors>
```

Jeżeli definicja akcji nie wyszczególnia użycia innego stosu, używany jest domyślny stos zdefiniowany poprzez:

```
<default-interceptor-ref name="loginStack"/>
```

Na poniższym schemacie przedstawiona została współpraca interceptorów i akcji *(Schemat 4*).



Schemat 4 Współpraca interceptorów i akcji

Źródło: Apache Struts 2 On-line Documentation http://struts.apache.org/2.x/docs/interceptors.html

4.4.6. Biblioteka tagów JSP

Struts2 udostępnia bibliotekę tagów jsp, które można wykorzystać do budowy stron. Biblioteka tagów zawiera elementy graficzne pojawiające się na stronie wynikowej, jak również elementy odpowiedzialne za sterowanie.

Tagi Struts2 posiadają wbudowany mechanizm zapytań Object-Graph Navigation Language (OGNL). Umożliwia on pobieranie danych z obiektu akcji, sesji HTTP, wywołania lub kontekstu aplikacji. Mechanizm języka przegląda kolejno zakresy widoku wymienione powyżej i pobiera wartość z pierwszego obiektu, który zostanie dopasowany do wzorca.

4.4.7. Budowa interfejsu Tiles2

Działająca na serwerze aplikacja komunikuje się z użytkownikiem poprzez przeglądarkę internetową, a co za tym idzie produkuje HTML. W niniejszej pracy opis poszczególnych stron oparty jest o JSP. Dla usprawnienia pracy została wykorzystana biblioteka Tiles2. Zastosowana biblioteka Tiles2 ułatwia utworzenie szablonów, które dla konkretnej strony mogą zawierać dane ogólne np. menu, nagłówek, stopka pochodzące z szablonu, jak i treść strony różną dla poszczególnych stron włączoną z innego pliku. Definicja takiej strony i przykładowe zastosowanie wygląda następująco:

```
<tiles:insertDefinition name="Login" />
```

W miejscu wystąpienia powyższego zapisu w pliku JSP zostanie wstawiona treść wynikająca z definicji zawartej w pliku ‘tiles.xml’. Definicja ta może przyjąć następującą postać:

```
<definition name="Login" template="/site/tiles/Layout_t.jsp">
  <put-attribute name="title"  value="STRONA LOGOWANIA"/>
  <put-attribute name="header" value="/site/Header.jsp"/>
  <put-attribute name="body"   value="/site/Login.jsp"/>
  <put-attribute name="footer"   value="/site/Footer.jsp"/>
</definition>
```

Do wygenerowania HTML zostanie użyty wzorzec strony ‘Layout_t.jsp’, który w konkretnych miejscach zostanie uzupełniony wartościami wskazanymi jako atrybuty.

Miejsca we wzorcu strony (tutaj /site/tiles/Layout_t.jsp), które zostaną uzupełnione danymi wynikającymi z konfiguracji, są oznaczone następującymi zapisami:

- <tiles:getAsString name="title"/>

Atrybut o nazwie ‘title’ zostanie wykorzystany bezpośrednio jako napis;

- <tiles:insertAttribute name="body"/>

Atrybut o nazwie ‘body’ zostanie wykorzystany jako atrybut, który należy dalej przetwarzać. W tym wypadku atrybut jest plikiem jsp.

4.4.8. Sprawdzanie poprawności wprowadzonych danych

W każdym oprogramowaniu serwerowym, które przyjmuje od użytkownika dane wpisywane w formularzach strony www, istnieje zapotrzebowanie na sprawdzanie poprawności wprowadzonych wartości, czyli walidację danych. Rozwiązanie tego problemu w Struts2 zostało opracowane na kilka sposobów. Pierwszym i najbardziej intuicyjnym jest uzupełnienie akcji o metodę validate(), która przeprowadzi walidacje. Pola klasy akcji można również uzupełnić odpowiednimi adnotacjami. Jednak korzystając z tych mechanizmów utracimy możliwość rozdzielenia sposobu walidacji od kodu aplikacji. Dobrym sposobem na takie rozdzielenie jest utworzenie pliku xml o nazwie takiej, jak nazwa klasy akcji, uzupełnionym o ‘-validation.xml’. Rozwiązanie takie udostępnia Struts2. Co ważne i ciekawe, w sytuacji, gdy dana klasa akcji dziedziczy po innej klasie akcji, czyli przejmuje jej pola i metody, sposób walidacji pól pochodzących z klasy bazowej zostanie zachowany. Innymi słowy, jeżeli klasy akcji dziedziczą po sobie, wówczas sposoby walidacji również dziedziczą po sobie, najczęściej się uzupełniając.

Przykładowy zapis z pliku ‘Login-validation.xml’:

```
<field name="username">
  <field-validator type="requiredstring">
    <message key="To pole musi być wypełnione"/>
  </field-validator>
</field>
```

Typ walidacji w przykładzie jest ustalony jako ‘requiderstring’ co oznacza, że pole oznaczone tak zdefiniowanym walidatorem nie powinno pozostać puste. Przy użyciu

tego walidatora sposób wypełnienia nie jest sprawdzany. Struts2 udostępnia kilka wbudowanych typów walidacji, tzw. walidatorów, w tym takich, których działanie oparte jest o wyrażenia regularne. Lista walidatorów nie jest zamknięta i może zostać uzupełniona przez programistę.

4.5. BAZA DANYCH I JNDI

Dla zapewnienia trwałości danych dostarczonych do tego oprogramowania konieczne jest wykorzystanie bazy danych. Odpowiednią bazą danych będzie PostegreSql, który oferuje transakcyjne przetwarzanie danych. Przy oprogramowaniu działającym dla wielu użytkowników oraz wielowątkowym ma to duże znaczenie. Aby zapewnić odpowiednią wydajność połączenia tego oprogramowania z bazą danych, wykorzystana powinna zostać pula połączeń. Serwer aplikacji Tomcat umożliwia zdefiniowanie dostępu do danych jako JNDI DataSource, który zostanie zdefiniowany jako usługa Java Naming and Directory Interface (JNDI).

Definicja źródła danych Tomcat JNDI DataSource może wyglądać następująco:

```
<Context>
<Resource name="jdbc/postgres" auth="Container"
driverClassName="org.postgresql.Driver"
factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
logAbandoned="true" maxActive="20" maxIdle="10" maxWait="1000"
removeAbandoned="true" removeAbandonedTimeout="60"
type="javax.sql.DataSource"
url="jdbc:postgresql://localhost:5432/formularze_mgr"
username=""
password=""/>
</Context>
```

Powyższa definicja określa sposób połączenia do bazy danych oraz parametry pracy niezbędne do nawiązania połączenia i wydajnej pracy oprogramowania zeń korzystającego.

Rozwiązanie takie umożliwia podłączenie lub zmianę serwera bazy danych bez zmiany kodu programu. Dodatkowo PostgreSql umożliwia utworzenie klastra, co zwielokrotni moc obliczeniową dostępną dla bazy danych i może okazać się przydatne przy dużej liczbie użytkowników. Pula połączeń jest bardzo przydatnym elementem oprogramowania, gdyż ułatwia zarządzanie wieloma połączeniami do

bazy. Jednocześnie limituje liczbę połączeń, co zapobiega przeciążeniu serwera, na którym uruchomiona jest baza. Ważnym elementem funkcjonowania puli jest fakt, iż nawiązane połączenia nie są przerywane lecz są przygotowane w puli do ponownego wykorzystania. Nawiązywanie połączenia jest procesem stosunkowo długotrwałym, dlatego wykorzystanie puli połączeń skutkuje przyspieszeniem pracy oprogramowania.

4.6. GOOGLE WEB TOOLKIT

Google Web Toolkit (GWT) jest to narzędzie, które wspomaga tworzenie rozbudowanych interfejsów użytkownika działających w przeglądarce w oparciu o JavaScript. Ułatwia budowę aplikacji AJAX umożliwiających asynchroniczną komunikację użytkownika z serwerem. Dzięki takiemu rozwiązaniu, w przypadku wykonywania działania z wykorzystaniem AJAX, nie jest konieczne przeładowywanie całego dokumentu, a jedynie jej wybranego fragmentu. Przyczynia się to do sprawniejszego i płynniejszego działania aplikacji. Narzędzie GWT może zostać wykorzystane do stworzenia części oprogramowania serwerowego odpowiedzialnego za umożliwienie wypełniania formularzy online. Decyzję taką warto uargumentować następującymi zaletami prezentowanego narzędzia:

- Pisanie kodu w Javie 5 z wykorzystaniem typów generycznych, autoboxingu etc.;
- Wygenerowany javascript działa we wszystkich przeglądarkach;
- Możliwości wykorzystania elementów środowiska programistycznego jak podpowiadanie składni, sprawdzenie poprawności syntaktycznej i semantycznej, refaktoring;
- Możliwość wykorzystania kodu Java stworzonego w innych częściach aplikacji;
- Debugowanie programu w języku Java za pomocą środowiska programistycznego.

Dodatkowym argumentem przemawiającym na korzyść tego narzędzia jest fakt, iż jest ono udostępniane na zasadach open source (na licencji Apache 2.0). Szeroko znane przykłady zastosowania GWT to Gmail czy też aplikacja Google Maps.

Jest to narzędzie, które umożliwia tworzenie skomplikowanych i dynamicznych formularzy na stronach www. Pomysł na tego typu produkt nie jest rewolucyjny, gdyż produkty o podobnej funkcjonalności występowały już wcześniej, np. projekt Echo. Powodem, dla którego GWT zostało zastosowane w tym prototypie, jest jego kompleksowość i ciągłe rozwijanie przez firmę Google, co zapewnia odpowiednią jakość takiego rozwiązania. Zasadą działania i ogromną zaletą tego produktu jest fakt, iż formularze występujące na stronach www można oprogramować w języku Java obiektowo, w sposób podobny do pisania aplikacji okienkowych z wykorzystaniem biblioteki Swing. Kod Javy jest następnie kompilowany do Java Script w sposób, który umożliwia uruchomienie w dowolnej przeglądarce. Warto tutaj zaznaczyć, że przeglądarki często różnią się między sobą w sposobie interpretacji Java Script.

4.7. BIBLIOTEKA APACHE HTTPCLIENT

Wykorzystanie biblioteki HTTPClient (patrz [4]) w Programie eksportującym dane na serwer (będącym jednym z wyników niniejszej pracy) umożliwi kompleksowa obsługę połączenia HTTP. Biblioteka jest implementacją standardu połączenia HTTP 1.0 i 1.1. Zapewnia ona obsługę standardowych metod HTTP takich jak:

- GET;
- POST;
- PUT;
- DELETE;
- HEAD;
- OPTIONS;
- TRACE.

Biblioteka pozwala również na transmisję z wykorzystaniem szyfrowanego protokołu HTTPS. Wszystkie wymienione wyżej zalety powodują, iż zastosowanie tej biblioteki umożliwi zbudowanie oprogramowania odpornego na zmieniające się wymagania projektu. Dodatkowo zapewniają one bezproblemowe uruchomienie oprogramowania w nieznanej w momencie projektowania architekturze sieciowej klienta w bezpieczny sposób.

4.8. iTEXT

iText jest biblioteką umożliwiającą automatyczne tworzenie plików PDF. Umożliwia dynamiczne tworzenie i/lub operowanie dokumentami PDF, przez co usprawnia możliwości aplikacji webowej, jak również aplikacji innego rodzaju. iText nie jest narzędziem stosowanym przez użytkownika końcowego, ale wbudowywanym w aplikację celem zautomatyzowania procesów tworzenia dokumentów PDF i operowania nimi. Przykładowo stosowanie iText zalecane jest w następujących sytuacjach:

- Ze względu na czas bądź rozmiar, dokumenty PDF nie mogą być tworzone w sposób ręczny;
- Zawartość dokumentu jest oparta bądź obliczana na podstawie danych dostarczonych przez użytkownika;
- Zawartość wymaga dostosowania do własnych potrzeb (potrzeb użytkownika) lub personalizacji;
- Zawartość pliku PDF wymaga udostępniania w środowisku webowym;
- Dokumenty są tworzone w trybie procesu wsadowego (z ang. *batch mode*), czyli za pomocą skryptu zawierającego instrukcje oraz dane tworzące zawartość pliku.

Bibliotekę iText można wykorzystać do:

- Udostępnienia plików tworzonych “w locie” za pomocą przeglądarki;
- Tworzenia dynamicznych dokumentów z plików XML bądź baz danych;
- Wykorzystywania wielu interaktywnych cech plików formatu PDF;
- Dodawania zakładek, numerów stron, znaków wodnych, itp.;
- Rozdzielania, łączenia (konkatenacji) i operowania stronami plików PDF;
- Automatyczne wypełnianie formularzy PDF;
- Dodawanie cyfrowych podpisów do plików PDF.

Klasy iText są przydatne po pierwsze w sytuacji, gdy istnieje potrzeba utworzenia niezależnych od platformy dokumentów tylko do odczytu, zawierających tekst, listy, tabele oraz obrazy. Po drugie, ich zastosowanie jest celowe, gdy mają zostać dokonane specyficzne operacje na istniejących dokumentach PDF. Zastosowanie biblioteki jest szczególnie przydatne w połączeniu z Servletami opartymi na technologii Java. Dostępny jest również port .NET iTextSharp (napisany w C#) służący połączeniu możliwości iText z Microsoft .NET .

5. ZAKRES FUNKCJONALNOŚCI APLIKACJI

Podstawową funkcją, którą z założenia powinno pełnić oprogramowanie będące przedmiotem niniejszej pracy, jest dostarczenie kompleksowego rozwiązania umożliwiającego tworzenie i wypełnianie formularzy elektronicznych utworzonych na podstawie zdjęcia lub skanu formularza papierowego. Przy tworzeniu oprogramowania duży nacisk został położony na prostotę jego obsługi, tak aby umożliwiała ona użytkownikowi wygodną i intuicyjną pracę nad przygotowaniem formularzy. Jednym z czynników, który przy tworzeniu przedmiotowego oprogramowania został uznany za mający istotny wpływ na łatwość jego obsługi, jest interfejs. Założono, iż interfejs powinien być złożony z elementów znanych użytkownikom oraz działających w sposób przewidywalny. Co więcej, samo oprogramowanie nie powinno zachowywać się w sposób znacząco inny niż oprogramowanie znane przez użytkownika. Kolejna wytyczna odnośnie funkcjonalności utworzonej aplikacji obejmowała przygotowywanie formularzy na możliwie dużym obszarze roboczym. Edytor formularzy powinien pozwolić na zapisywanie i odczytywanie plików przygotowanych wcześniej formularzy oraz umożliwiać ich edycję.

Podstawową funkcją serwera aplikacji będzie generowanie plików PDF na podstawie dostarczonych danych opisujących formularz i danych go wypełniających. Ponadto serwer aplikacji powinien umożliwić import pliku formularza wygenerowanego za pomocą Edytora Formularzy, jak również danych służących do wypełnienia oznaczonych pól.

Nieodzowną podstawą przy projektowaniu budowy części serwerowej aplikacji było założenie o możliwości jej rozbudowy w sposób niewymagający ingerencji w już istniejące części oprogramowania. Zgodnie z kolejnym wymogiem postawionym tworzonemu oprogramowaniu, zadaniem serwera jest zapewnienie trwałości danych wprowadzonych przez użytkownika poprzez wykorzystanie w tym celu bazy danych.

W odniesieniu do Programu eksportującego dane stwierdzono, iż należy zaimplementować go w taki sposób, aby po uprzedniej konfiguracji działał bezobsługowo, jak również w sposób niezauważalny dla użytkownika. Przewiduje się, iż w najbardziej prawdopodobnym sposobie użytkowania oprogramowanie powinno uruchamiać się razem z systemem operacyjnym i działać aż do momentu wyłączenia komputera.

5.1. EDYTOR FORMULARZY

Podstawową funkcjonalnością oferowaną przez Edytor formularzy ma być możliwość utworzenia formularza w sposób, który nie będzie powodował zakłopotania u obsługującego program użytkownika. Celem osiągnięcia zarówno prostoty obsługi, jak i podobieństwa w logice działania do analogicznych produktów na rynku, przy tworzeniu Edytora formularzy wykorzystana została biblioteka Swing dostarczana wraz z Java Runtime Environment. Przyjęte rozwiązanie pozwoli uniknąć trudności w uruchamianiu aplikacji na komputerze użytkownika aplikacji. Przy wyborze biblioteki Swing kierowano się również tym, iż oferuje ona komponenty takie jak: przyciski, pola edycyjne, tabelki, elementy interfejsu odpowiedzialnego za interakcję z użytkownikiem. Dodatkową zaletą tej biblioteki jest dostępność komponentów służących grupowaniu innych komponentów w tak zwanych kontenerach (przykładowo panele). Warto zaznaczyć, iż biblioteka Swing została napisana w sposób umożliwiający wykorzystanie akceleracji sprzętowej przy wyrysowywaniu elementów interfejsu na ekranie, co powinno zapobiegać migotaniu obszaru roboczego w trakcie pracy.

Dla zapewnienia spójności danych opracowywanych za pomocą Edytora formularzy został utworzony model danych, który dba o przechowywane informacje. Przede

wszystkim model ten jest wykorzystywany do przechowywania niezbędnych informacji o utworzonym formularzu, takich jak jego nazwa, nazwa pliku tła, jak również informacji o polach danych, ich kolejności oraz liczebności. Zadaniem modelu danych jest również dbałość o aktualność danych pokazywanych w powiązanych z nim komponentach graficznych interfejsu. Operacje udostępniane przez model danych są zaprojektowane w sposób umożliwiający współpracę wielu elementów interfejsu. Z istotnych cech przyjętego rozwiązania warto nadmienić, iż model danych jest w tym wypadku obiektem, który dba o spójność informacji. Jest on również odpowiedzialny za propagację zmian i informowanie innych elementów interfejsu o zmianach zachodzących w danych.

Z punktu widzenia zapewnienia możliwości prostego tworzenia formularzy wskazane jest, aby Edytor formularzy oferował następujące funkcje:

- Wczytanie i wyświetlenie pliku tła, czyli zdjęcia formularza papierowego;
- Wykreślenia za pomocą wskaźnika myszki obszarów, gdzie powinny znaleźć się dane formularza;
- Przesuwanie za pomocą myszki pól danych w obszarze roboczym;
- Zmiana rozmiarów pól danych;
- Ustalanie nazw pól danych;
- Zmiana rozmiarów i położenia pól danych za pomocą klawiatury;
- Usuwanie zbędnych pól danych;
- Ustalanie nazwy formularza;
- Zapisywanie danych do pliku;
- Odczyt danych z pliku;
- Zmiana rozmiarów okna aplikacji;
- Zmiana proporcji podziału okna na obszar roboczy i inspektor obiektów.

Z kolei dla zapewnienia wygody obsługi powinien zostać opracowany taki sposób tworzenia formularzy, który spełniałby dodatkowe wymagania takie jak:

- Wykrywanie ruchów myszki nad polem danych;
- Wykrywanie ruchów myszki nad krawędzią pola danych;
- Reagowanie na otrzymanie lub utratę kursora/sterowania przez obszar roboczy, jak również przez poszczególne pola danych.

Najlepszym sposobem zapewnienia odpowiedniego zachowania aplikacji biorąc pod uwagę powyższe wymagania dotyczące funkcji obszaru roboczego oraz obsługiwanych komunikatów pochodzących od użytkownika będzie rozszerzenie oferowanych możliwości biblioteki Swing. Zdecydowano o utworzeniu własnego komponentu dziedziczącego po podstawowym komponencie biblioteki Swing JComponent. Niezbędne jest, aby nowy komponent wyrysowujący obszar roboczy był zintegrowany z modelem danych. Dla zapewnienia wystarczającej wydajności i odpowiedniego efektu graficznego, za zasadne uznano wykorzystanie mechanizmu podwójnego buforowania. Przez efekt graficzny rozumiane są działania takie jak przeciąganie elementów, rysowanie pól czy zmiana rozmiarów pól, gdy w tle jest wyświetlony obraz tła. Mechanizm podwójnego buforowania polega na tym, że stosunkowo długotrwałe rysowanie odbywa się w obszarze pamięci operacyjnej odwzorowującym ekran w sposób niewidoczny dla użytkownika. Następnie gotowy już obraz jest kopiowany i uwidaczniany w odpowiednim miejscu na ekranie. Podwójne buforowanie powinno skutkować znaczącym zmniejszeniem migotania obrazu.

Jedną z podstawowych funkcji, które z założenia powinny być dostępne za pomocą tego oprogramowania, jest zapis i odczyt stanu prac z pliku. Zastosowano podejście, iż funkcja zapisu utworzonego formularza umożliwia wybranie zarówno miejsca zapisu, jak i nazwy pliku, w którym zostaną zapisane dane. Niezbędne jest, aby dane zapisane do pliku były w formacie zrozumiałym dla serwera aplikacji. Ponadto format zapisu powinien zezwalać na łatwą rozbudowę. Oczekuje się, iż odczyt zapisanych danych umożliwi wybór pliku z drzewa katalogów. Wskazane jest, by po wczytaniu danych program znajdował się w stanie pozwalającym na dalszą pracę nad formularzem.

Następnie zapisany plik zawierający przygotowany formularz zostanie zaimportowany na serwer, który jest kolejnym elementem prototypu systemu utworzonego w ramach niniejszej pracy.

5.2. SERWER

Zasadniczym celem, jaki powinien zostać spełniony przez serwerową część prototypu, jest wygenerowanie pliku PDF na podstawie dostarczonych informacji. Do informacji wykorzystywanych przez serwer zalicza się następujące elementy:

- Opis formularza utworzonego w Edytorze Formularzy;
- Dane służące do wypełnienia tego formularza, dostarczone za pomocą programu do eksportu danych do serwera, wprowadzone online w przeglądarce lub poprzez import przygotowanego pliku na stronie do tego służącej.

Architektura serwera powinna zostać oparta o nowoczesny szkielet programistyczny zapewniający zarówno możliwość rozbudowy, jak i udostępniający przystępny sposób konfiguracji. Uwzględniając powyższe czynniki stwierdzono, iż właściwym rozwiązaniem będzie Struts2 opierający się w znacznym stopniu o framework WebWork oraz korzystający z doświadczeń zdobytych przy pracach nad Struts w pierwszej wersji. Ze względu na korzyści związane z zastosowaniem Struts2 zdecydowano, iż jest to właściwe rozwiązanie dla tego typu programu. W przypadku zaistnienia takiej potrzeby, możliwe jest dołączenie bibliotek rozszerzających możliwości Struts2 za pomocą mechanizmu pluginów. Ponadto oferuje on mechanizm pozwalający wykonywać czynności przed (z ang. *preprocessing*) lub po wykonaniu (z ang. *postprocessing*) akcji odpowiedzialnej za logikę aplikacji - mechanizm interceptorów. Interceptory zgrupowane w stos zapewniają często obsługę operacji nie wywodzących się bezpośrednio z logiki biznesowej aplikacji, a niezbędnych dla zapewnienia bezpieczeństwa dostępu lub wygody programisty.

Część serwerowa opracowywanego oprogramowania posiada moduł umożliwiający zarządzanie dostępem użytkowników.

Dynamiczne generowanie stron internetowych powinno być wspomagane narzędziem, które umożliwi podział zawartości stron na moduły, jak również prostą konfigurację i wielokrotne użycie elementów powtarzających się na stronach. W tym zakresie pomocna będzie biblioteka Tiles2, która oferuje możliwość wykorzystania szablonów. Szablon taki może zostać wyposażony w miejsca uzupełniane treścią w sposób dynamiczny, jak i w treści statyczne niezmienne, występujące na każdej stronie. Definicja strony Tiles2 zawiera informacje o treści, którą powinny zostać wypełnione oznaczone miejsca szablonu.

5.3. PROGRAM EKSPORTUJĄCY DANE

Głównym zadaniem Programu Eksportującego Dane jest eksport danych na serwer, a następnie pobranie wydruku w postaci pliku PDF. Dane mogą zostać wytworzone za pomocą oprogramowania zewnętrznego i udostępnione w postaci pliku XML omawianemu programowi.

Do wykonania tego zadania potrzebna jest biblioteka, która pozwoli na połączenie z serwerem HTTP i wysłanie do niego danych. Takie wymogi spełnia biblioteka tworzona w ramach organizacji Apache - HTTPClient. Dzięki zastosowaniu tej biblioteki możliwe będzie nawiązanie połączenia i wysłanie danych opakowanych jako parametry metody HTTP POST. Co więcej, biblioteka ta umożliwia również pobieranie plików w postaci binarnej, jakimi są dokumenty w formacie PDF.

Konfiguracja aplikacji powinna udostępniać funkcję określania katalogu do monitorowania, a także nazwy i hasła użytkownika zdefiniowanego na serwerze, na którego konto zostanie przeprowadzony eksport itp. Dane konfiguracyjne należy przechowywać w sposób trwały jako plik na dysku twardym. W tym wypadku warto wykorzystać możliwości oferowane przez J2SE w zakresie przechowywania i obsługi parametrów aplikacji. Takie możliwości zapewnia klasa Properties ze standardowej biblioteki Java. Klasa Properties pozwala na zapisanie pliku konfiguracyjnego w postaci XML, jak również jego późniejsze wczytanie.

W normalnym trybie działania program eksportujący dane powinien pozostać niewidoczny dla użytkownika. Przy tworzeniu niniejszego oprogramowania zdecydowano, iż najlepszym rozwiązaniem będzie schowanie programu w „Trayu”, czyli obok zegara systemowego w systemie operacyjnym Windows. Dostęp do okna programu powinien być zapewniony poprzez kliknięcie w ikonę programu. W celu zapewnienia takiej funkcjonalności trzeba wykorzystać wbudowane mechanizmy JavaSE 6 lub biblioteki dodatkowe wraz z Java 5. Biblioteka standardowa Javy 6 udostępnia klasę TrayIcon i SystemTray, które umożliwiają oprogramowanie żądanego zachowania aplikacji.

6. PREZENTACJA PROTOTYPÓW

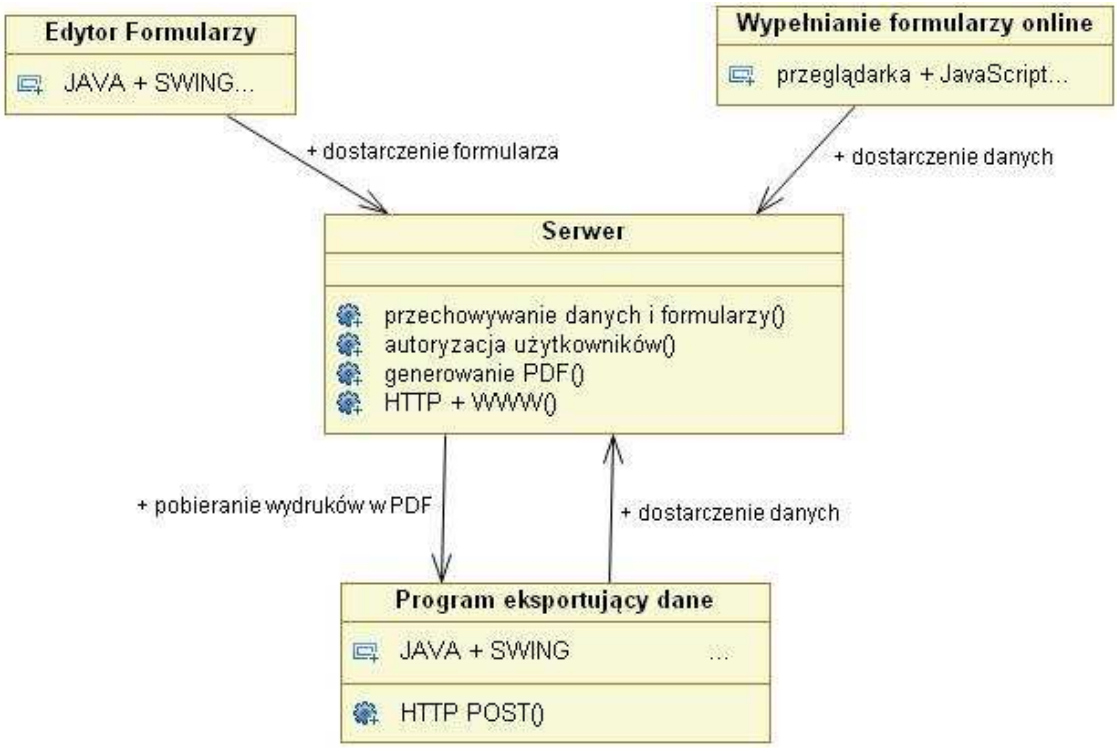
Prototypy opracowane w ramach niniejszej pracy mają zapewnić kompleksową obsługę tworzenia i wypełniania formularzy. Zastosowanie tego oprogramowania powinno skutkować zmniejszeniem ilości pracy wykonywanej przez pracowników, unifikację wyglądu formularzy, zautomatyzowanie wypełniania formularzy danymi, a także archiwizację tych danych.

Mając na względzie optymalne wykorzystanie nowoczesnych technologii tworzenia oprogramowania, zakładana dla tej pracy funkcjonalność została podzielona pomiędzy trzy aplikacje, tj.:

- Edytor formularzy;
- Serwer formularzy umożliwiający wygenerowanie wydruków, jak również zarządzanie użytkownikami i magazynowanie dostarczonych danych;
- Program eksportujący dane do serwera i pobierający gotowe wydruki.

Dwie pierwsze z wyżej wymienionych aplikacji zostały opracowane jako pogramy uruchamiane na komputerze klienta korzystające z funkcjonalności dostępnej za pomocą biblioteki standardowej Java Swing. Natomiast trzecia aplikacja jest aplikacją serwerową komunikującą się z użytkownikiem za pomocą przeglądarki internetowej. W części serwerowej można wyszczególnić podprojekt wykonany w innej technologii, który jest odpowiedzialny za wypełnianie formularzy w przeglądarce internetowej. Do stworzenia tego podprojektu wykorzystano narzędzie Google Web Toolkit.

Wymienione powyżej aplikacje współpracują ze sobą w celu zapewnienia pełnej obsługi tworzenia i wypełniania formularzy. Przebieg danych w ramach tego systemu pokazuje poniższy schemat przepływu danych *(Schemat 5*).



Schemat 5 Schemat przepływu danych w systemie

Źródło: Opracowanie własne

Wszystkie programy wchodzące w skład tego zestawu zostały utworzone za pomocą środowiska programistycznego Eclipse w wersji 3.3.2 przystosowanego do tworzenia oprogramowania z użyciem JEE. Podprojekt służący wypełnianiu formularzy w przeglądarce internetowej, poza Google Web Toolkit wykorzystuje również, plugin Cypal Studio do środowiska Eclipse, ułatwiające prace programistyczne. Plugin ten umożliwia usprawnienie procesu tworzenia, kompilowania i debugowanie kodu programu.

6.1. MODEL DANYCH OPISUJĄCY FORMULARZE

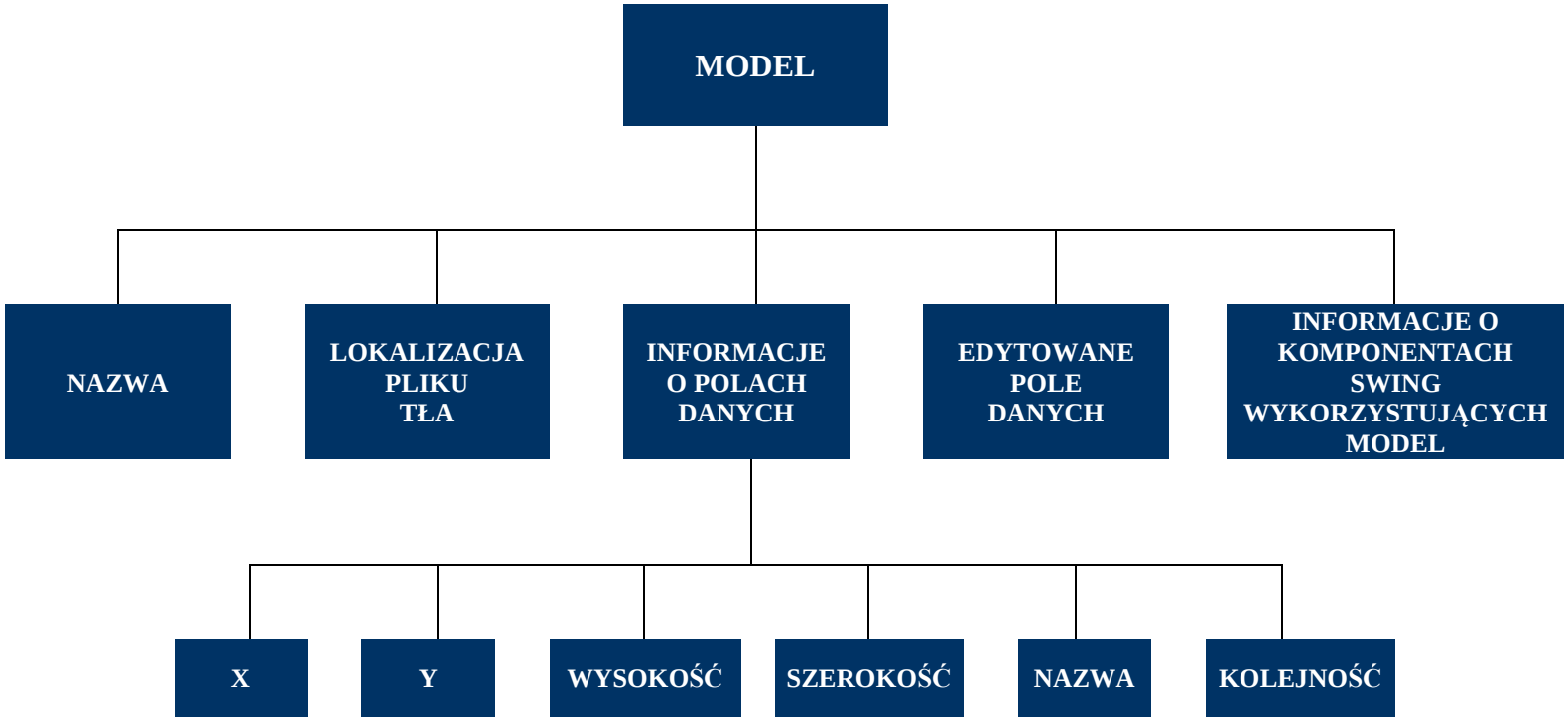
W związku z faktem, iż funkcjonalność oprogramowania jest ściśle określona i odpowiada za operacje związane z tworzeniem i wypełnianiem formularzy, możliwe było wydzielenie modelu danych. Wspólny dla wszystkich aplikacji model danych, przechowujący informacje opisujące formularz, gwarantuje spójność informacji oraz jednolity sposób dostępu. Jednocześnie zastosowanie modelu danych zapewnia wielokrotne użycie raz utworzonego kodu. Jest to istotne z punkt widzenia programisty, gdyż umożliwia zmniejszenie nakładów pracy potrzebnych przy tworzeniu i późniejszym utrzymaniu lub rozbudowie zakresu programu.

Zastosowany model danych przechowuje następujące informacje:

- Nazwa modelu;
- Lokalizacja pliku graficznego zawierającego tło formularza;
- Informacje sumaryczne o polach danych;
- Wysokość, szerokość, współrzędne lewego górnego rogu każdego pola danych;
- Nazwy nadane polom danych;
- Informacje o aktualnie zaznaczonym, a co za tym idzie aktualnie edytowanym polu danych;
- Informacje o kolejności rysowania pól danych.

Model danych jest używany jako element centralny wszystkich zmian danych. Model przechowuje informacje o komponentach, które wykorzystują dane w nim zawarte i informuje je o wszelkich zmianach. Propagacja informacji o zmianach skutkuje odświeżeniem prezentowanych danych we wszystkich komponentach.

Na kolejnej stronie zaprezentowany został schemat budowy modelu danych (*Schemat 6*).



Schemat 6 Schemat budowy modelu danych

Źródło: Opracowanie własne

Propagacja informacji o zmianach danych jest możliwa za pomocą mechanizmu PropertyChangeListener. Mechanizm ten opiera się na założeniu, iż obiekt klasy zawierającej informacje wykorzystywane w innych miejscach aplikacji może w sposób aktywny powiadamiać o ich zmianie. Pakiet biblioteki standardowej java.beans oferuje klasę PropertyChangeSupport implementującą przedstawione założenie. Pakiet java.beans rozszerza możliwości podstawowych klas języka Java, tzw. POJO (z ang. *Plain Old Java Object*), czyli klas Java Beans – ziaren Javy. Warto zwrócić szczególną uwagę na fakt, iż JavaBeans nie ma wielu cech wspólnych z technologią Enterprise Java Beans opisaną w specyfikacji Java Enterprise Edition.

W tym przypadku model danych zawiera obiekt typu PropertyChangeSupport. Obiekt ten przechowuje listę obiektów zainteresowanych odbieraniem komunikatów o zmianie wartości poszczególnych pól.

Model zawiera również dwie metody addPropertyListener oraz removePropertyListener odpowiedzialne za manipulację listą „słuchających" obiektów. Zastosowanie mechanizmu omówionego powyżej wygląda następująco:

```
private PropertyChangeSupport pcs = new PropertyChangeSupport(this);

public void addPropertyListener(PropertyChangeListener obser) {
  if(obser!=null){
    pcs.addPropertyChangeListener(obser);
  }
}
public void removePropertyListener(PropertyChangeListener obser) {
  if(obser != null){
    pcs.removePropertyChangeListener(obser);
  }
}
```

Z powyższego zapisu wynika, iż dodawane do listy obiekty klas muszą implementować interfejs PropertyChangeListener, który obliguje je do implementacji metody:

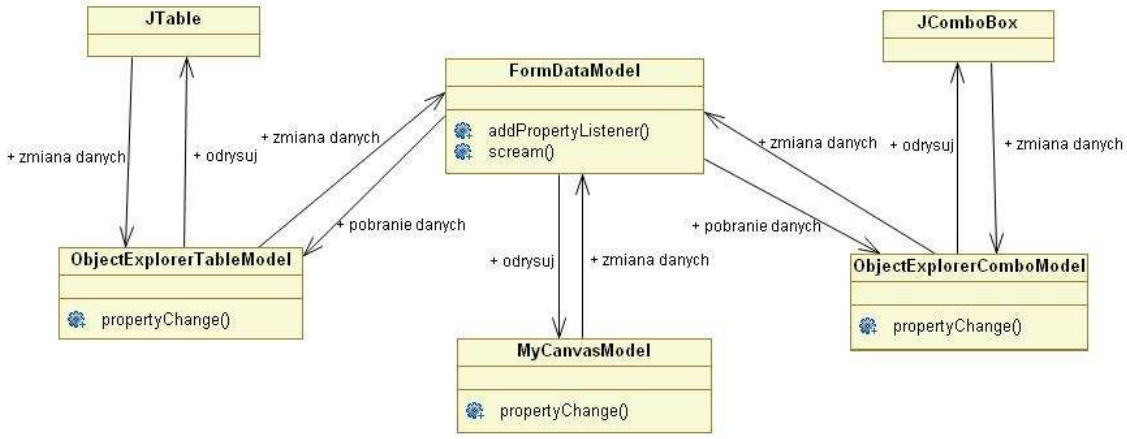
```
void propertyChange(PropertyChangeEvent evt);
```

Parametrem tej metody jest PropertyChangeEvent, który zawiera wszystkie informacje potrzebne do podjęcia decyzji odnośnie tego, czy konieczna jest reakcja na przychodzące zdarzenie.

Model danych powiadamia o zmianie parametrów wybranego pola danych. Wysłanie komunikatów o zmianie następuje w momencie wywołania następującej metody:

```
private void scream() {
  pcs.firePropertyChange("selectedAreaId",null,selectedAreaId);
}
```

Schemat przepływu danych i informowania o ich zmianach przedstawiono na schemacie współpracy modelu danych z interfejsem Edytora Formularzy *(Schemat 7).*



Schemat 7 Schemat współpracy modelu danych z interfejsem Edytora Formularzy

Źródło: Opracowanie własne

6.2. EDYTOR FORMULARZY – MECHANIZMY KOMUNIKACJI

Edytor formularzy jest aplikacją napisaną w Java 5 Standard Edition. Wykorzystanie Javy zostało podyktowanie różnorodnością wykorzystywanych systemów operacyjnych przez potencjalnych użytkowników. Java umożliwia utworzenie oprogramowania niezależnego od systemu operacyjnego, na którym będzie ono uruchamiane – warunkiem wystarczającym jest obecność odpowiedniego środowiska uruchomieniowego Javy. Dodatkowo standardowa instalacja J2SE w postaci Java Runtime Environment zawiera bibliotekę Swing ułatwiająca tworzenie aplikacji okienkowych. Istotną cechą aplikacji okienkowych jest fakt, iż na ich wygląd nie będzie miało wpływu środowisko, w którym są uruchamiane.

6.2.1. Mechanizm Listener

Prezentowany na ekranie interfejs komunikuje się z użytkownikiem za pomocą klawiatury i myszy. Każde działanie użytkownika może zostać spożytkowane w konkretny sposób bądź opuszczone jako nieznaczące. Mechanizm dostarczany przez bibliotekę Awt i Swing umożliwia użycie zdefiniowanych metod komponentów i interfejsów odpowiedzialnych za nasłuchiwanie zdarzeń (*z ang. Listener*) pochodzących od użytkownika. Komponenty graficzne biblioteki Swing oferują podział zdarzeń zachodzących w programie m.in. na:

- MouseListener – przychodzące z myszki komputerowej;
- KeyListener – przychodzące z klawiatury.

oraz wiele innych.

Wykorzystanie mechanizmu MouseListener może wyglądać następująco:

```
Jpanel.addMouseListener( new MouseListener(){
          (...)
}
```

W momencie zaistnienia zdarzenia, które jest związane z obsługą myszy, zostanie wywołana klasa anonimowa stworzona na podstawie interfejsu MouseListener, która może podjąć działania obsługujące to zdarzenie. Interfejs MouseListener prezentuje się następująco (kod zaczerpnięty z kodu źródłowego języka Java wraz z komentarzami):

```
package java.awt.event;
public interface MouseListener extends EventListener {
    /**
     * Invoked when the mouse button has been clicked (pressed
     * and released) on a component.
     */
    public void mouseClicked(MouseEvent e);

    /**
     * Invoked when a mouse button has been pressed on a component.
     */
    public void mousePressed(MouseEvent e);

    /**
     * Invoked when a mouse button has been released on a component.
     */
    public void mouseReleased(MouseEvent e);

    /**
```

```
     * Invoked when the mouse enters a component.
     */
    public void mouseEntered(MouseEvent e);

    /**
     * Invoked when the mouse exits a component.
     */
    public void mouseExited(MouseEvent e);
}
```

Mechanizm KeyListener wygląda analogicznie do przedstawionego MouseListener. Zastosowanie tych mechanizmów umożliwia nawiązanie interakcji z użytkownikiem i zbudowanie interfejsu użytkownika.

6.2.2. Interfejs użytkownika

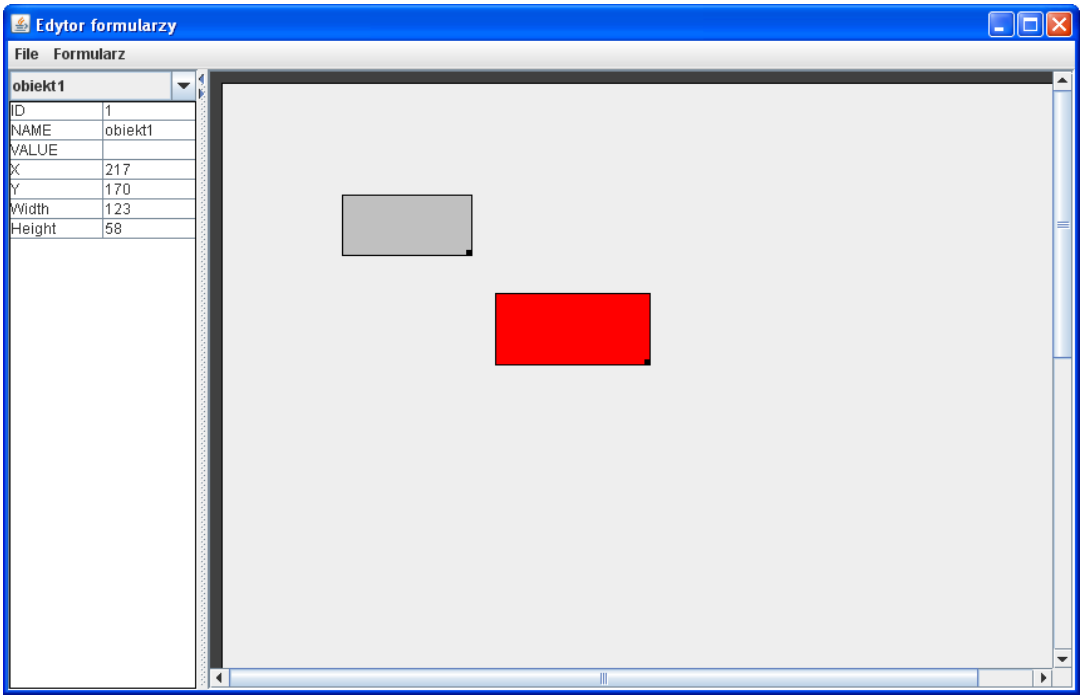
Jak już wspomniano, podstawową rolą Edytora formularzy jest dostarczenie kompleksowego i łatwego w obsłudze narzędzia, które umożliwiałoby utworzenie formularza w formie elektronicznej na podstawie zdjęcia lub skanu jego papierowej wersji. Wypełnienie postawionego przed tym programem zadania nie byłoby możliwe bez zaprojektowania intuicyjnego, przyjaznego użytkownikowi interfejsu.

Elementarnymi funkcjami programu, które zostały zaimplementowane i znalazły odbicie w interfejsie użytkownika są:

- Zaznaczanie nowego pola danych;
- Przesuwanie pola danych;
- Zmiana rozmiarów pola danych;
- Zaznaczanie edytowanego pola danych;
- Wyświetlenie danych edytowanego pola danych;
- Ustalanie nazwy modelu;
- Wybieranie pliku tła formularza;
- Zapis i odczyt pliku formularza.

Utworzony interfejs przedstawiono na obrazku na kolejnej stronie pracy *(Rysunek 8*). Okno aplikacji przedstawia elementy wchodzące w skład aplikacji. Widoczne są obszar roboczy z prawej strony z narysowanymi dwoma polami danych w tym jedno jest

aktywne i oznaczone kolorem czerwonym. Po lewej stronie ekrany widoczny jest inspektor obiektów przedstawiający dane opisujące zaznaczone pole danych. Na górze okna widoczne jest Menu użytkownika które zostało opisane poniżej.



Rysunek 8 Interfejs Edytora Formularzy

Źródło: Print Screen z programu Edytor formularzy

6.3. EDYTOR FORMULARZY – BUDOWA I OPIS DZIAŁANIA INTERFEJSU

6.3.1. Menu

Menu interfejsu udostępnia funkcje oprogramowania niedostępne z innych miejsc. Opis funkcji wraz z przypisaniem do poszczególnych poziomów menu Edytora formularzy został zawarty w poniższej tabeli (*Tabela 1*).

Tabela 1 Opis funkcji menu Edytora formularzy

Funkcje menu Edytora formularzy			
MENU	Funkcje menu		Opis funkcji
	Plik	Otwórz	Umożliwia otwarcie wcześniej zapisanego na dysku twardym pliku formularza. Dane odczytane z pliku są wczytywane do wcześniej wyczyszczonego modelu danych.
		Zapisz	Umożliwia zapisanie do pliku danego formularza przygotowanego w programie. Taki plik może zostać zapisany na serwerze w celu późniejszego wykorzystania do wygenerowania wydruku po wprowadzeniu niezbędnych danych.
		Wyjście	Wyjście z programu.
	Formularz	Wprowadź nazwę	Umożliwia wprowadzenie nazwy przygotowywanego formularza. Nazwa ta zostanie podpowiedziana użytkownikowi przy imporcie formularza na serwer. Nazwa formularza wprowadzona na serwerze umożliwia jego identyfikację i łączenie z wypełniającymi go danymi.
		Wybierz zdjęcie tła	Umożliwia wybranie pliku graficznego, który będzie stanowił tło formularza. Na podstawie zdjęcia lub skanu możliwe jest przygotowanie formularza poprzez zaznaczenie w odpowiednich miejscach pól do wprowadzenia danych. Wczytanie pliku graficznego spowoduje wyświetlenie jego zawartości w obszarze roboczym, przy czym zaznaczane pola danych będą widoczne. Nazwa pliku tła formularza zostanie podpowiedziana przez serwer w momencie importu formularza na serwer.

Źródło: Opracowanie własne

6.3.2. Inspektor obiektów

Kolejnym elementem interfejsu użytkownika Edytora formularzy jest widoczny w jego lewej części Object Inspector. Jego zasadnicze działanie polega na umożliwieniu ręcznej edycji parametrów pól danych. Składa się z połączonych funkcjonalnie ze sobą komponentów typu Combo i Table. Obiekt Combo zawiera listę nazw wyrysowanych

pól danych. Wybór elementu z tej listy spowoduje zaznaczenie wybranego elementu kolorem czerwonym w obszarze roboczym. Parametry tak wybranego elementu zostaną wyświetlone w tabeli poniżej listy wyboru. Warto nadmienić, iż parametry zawarte w tabeli mogą być edytowane. Jeżeli zmiana parametrów jest związana z właściwościami odpowiedzialnymi za rozmiar lub położenie edytowanego elementu, zostanie to uwidocznione poprzez zmianę wyglądu tego pola w obszarze roboczym.

6.3.3. Obszar roboczy

Obiektami odpowiedzialnym za wyrysowanie i podstawową obsługę użytkownika w tym programie są Canvas i Scrollpanel. Obiekt obszaru roboczego został umieszczony na JscrollPane, czego efektem jest możliwość dostosowania okna aplikacji do rozmiaru ekranu widocznego dla użytkownika. W momencie, gdy obszar roboczy nie mieści się w oknie, pojawiają się suwaczki odpowiedzialne za przesuwanie zawartości obszaru roboczego. W sytuacji, gdy obszar roboczy mieści się w przeznaczonym dla niego miejscu na szerokość, paski przewijania nie są widoczne, a dodatkowo następuje wyśrodkowanie poziome obszaru odpowiedzialnego za rysowanie poprzez zwiększenie marginesów w kolorze szarym.

6.3.4. Pionowy podział ekranu

Elementem interfejsu umożliwiającym dostosowanie go do preferencji użytkownika jest komponent JsplitPane pochodzący z biblioteki Swing. Jego zadaniem jest umożliwienie zmiany poziomych proporcji podziału okna na Object Inspector i obszar roboczy.

Działanie wyżej wymienionych elementów oraz ich wzajemne oddziaływanie na siebie zostało opisane poniżej.

6.3.5. Działanie i budowa interfejsu użytkownika

Podstawowym elementem wchodzącym w interakcję z użytkownikiem jest obszar roboczy odpowiedzialny za kreślenie pól, z którymi w przyszłości będą wiązane dane. W wyniku dążenia do spełnienia warunków, tj. zapewnienia wygody użytkownika przy jednoczesnym umożliwieniu nieskomplikowanej rozbudowy, zdecydowano się na

rozwiązanie, w którym obszar roboczy jest komponentem stworzonym poprzez dziedziczenie na podstawie podstawowej klasy komponentów JComponent z biblioteki Swing. Klasa komponentu jest odpowiedzialna za współpracę graficzną z użytkownikiem i w połączeniu z modelem danych pozwala na edycję pól danych. Utworzony komponent został nazwany MyCanvasModel. Komponent wywodzący się z klasy Swing oferuje możliwość osadzenia go w oknie za pomocą standardowych mechanizmów przewidzianych dla komponentów Swing. Komponent obsługuje następujące zdarzenia zachodzące w jego otoczeniu:

- Zmiana wielkości okna powoduje zmianę rozmiaru komponentu;
- Zmiana rozmiaru części okna przydzielonej dla obszaru roboczego skutkuje zmianą rozmiaru komponentu;
- Przesłonięcie i odsłonięcie przez inne okno powoduje odmalowanie zawartości;
- Minimalizacja okna oraz powrót do normalnego rozmiaru powoduje odmalowanie zawartości.

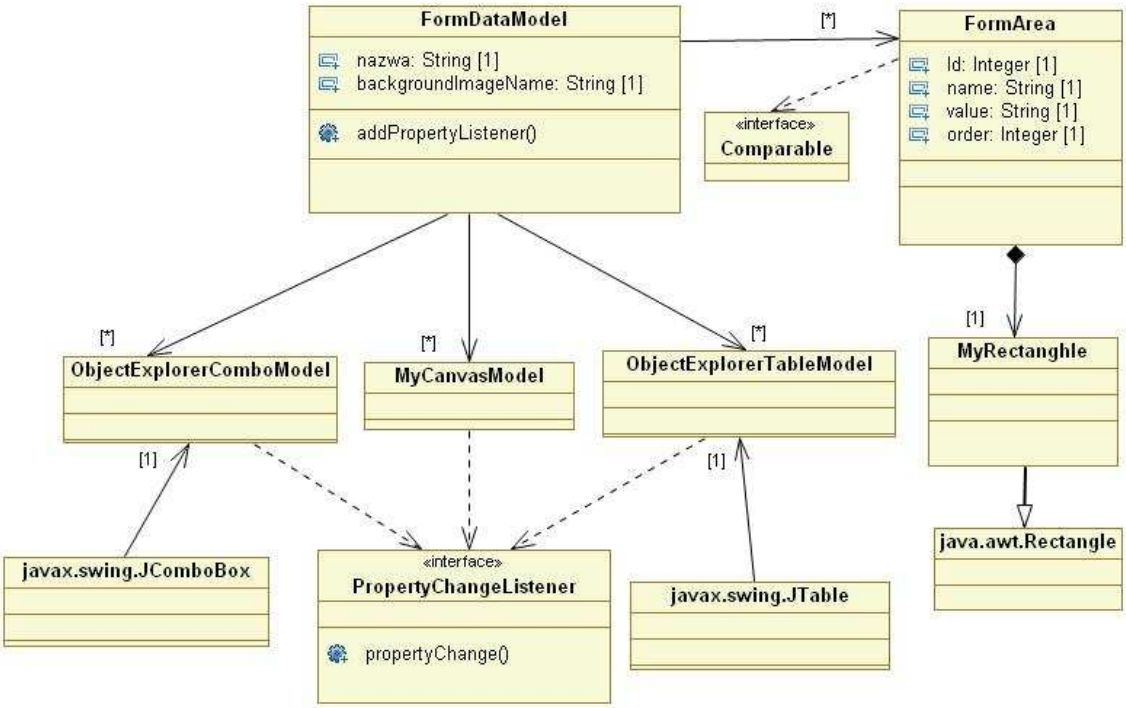
MyCanvasModel odbiera komunikaty generowane przez działanie użytkownika. Poniżej wyszczególnione zostały typy odbieranych komunikatów, które są obsługiwane przez MyCanvasModel:

- KeyListener – odbieranie komunikatów pochodzących z klawiatury;
- MouseListener – obsługa komunikatów pochodzących od myszy – statyczne;
- MouseMotionListener – obsługa komunikatów pochodzących od myszy – dynamiczne, ruch kursora z wciśniętym przyciskiem myszy (przeciąganie) lub bez;
- PropertyChangeListener – reakcja na zdarzenie pochodzące od modelu danych;
- FocusListener – aktywacja lub dezaktywacja komponentu, odpowiednik kursora.

Faktem wartym odnotowania jest możliwość umieszczenia obiektu MyCanvasModel we wnętrzu kontenera np. Panelu posiadającego paski boczne. Takie działanie może służyć do przesuwania treści gdy komponent MyCanvasModel nie mieści się w oknie. Fakt ten został wykorzystany w tym programie. W sytuacji gdy obszar roboczy jest większy niż obszar dostępny, w oknie pojawiają się paski przewijania pochodzące z JSplitPane. Wprowadzenie takiego rozwiązania nie wywołało konieczności wprowadzenia jakichkolwiek zmian w obiekcie reprezentującym obszar roboczy.

W programie prowadzona jest na bieżąco kontrola położenia kursora. Kiedy nad obszarem roboczym przesuwany jest kursor myszki, sprawdzane jest, czy w danym momencie nie znajduje się on nad polem danych. Położenie kursora nad polem danych spowoduje zmianę wyglądu kursora. Dzięki temu użytkownik otrzyma informację o operacji, która zostanie wykonana bądź rozpoczęta po przyciśnięciu lub przytrzymaniu klawisza myszy. Jednokrotne wciśnięcie klawisza myszy skutkuje zaznaczeniem i uaktywnieniem obszaru znajdującego się pod kursorem. Z kolei przytrzymanie klawisza myszy rozpocznie akcję przesuwania obszaru w nowe miejsce. Natomiast jeżeli kursor znajduje się w prawy dolnym rogu pola danych, wówczas zostanie zainicjowana operacja zmiany rozmiaru pola danych.

Współpraca elementów interfejsu jest możliwa dzięki centralnemu zarządzaniu danymi stworzonemu w oparciu o model danych. Mechanizm PropertyChangeSupport umożliwia przekazywanie informacji elementom interfejsu o zmianach danych. Poniżej przedstawiony schemat (*Schemat 8*) przedstawia sposób implementacji mechanizmu PropertyChangeSupport w połączeniu z modelem danych. Wspomniany schemat jest diagramem klas w notacji UML (ang. Unified Modeling Language), o którym można przeczytać w [5].



Schemat 8 Diagram klas wchodzących w skład modelu danych i mechanizmu PropertyChangeSupport

Źródło: Opracowanie własne

6.4. SERWER

Podstawową funkcją prototypu serwera jest generowanie wydruków na podstawie wcześniej przygotowanych formularzy oraz dostarczonych przez użytkownika danych. Założono, że sposobem niezależnym od systemu operacyjnego i nieprzysparzającym użytkownikowi problemów jest dostarczenie wydruku formularza w formie pliku PDF. Zaletą przyjętego rozwiązania jest fakt, iż taki plik może zostać otwarty w odpowiednim programie na komputerze użytkownika i wyglądać w sposób jednakowy niezależnie od zainstalowanego oprogramowania i środowiska operacyjnego. Tworząc oprogramowanie serwerowe dążono do tego, aby oferowało ono możliwość interakcji z użytkownikiem za pomocą przejrzystego interfejsu wyświetlanego jako strona internetowa w przeglądarce internetowej.

Oprogramowanie działające na serwerze, dostępnym za pomocą sieci komputerowej, musi w pierwszej kolejności charakteryzować się możliwością obsługi wielu

użytkowników jednocześnie, a co za tym idzie wystarczającą do ich obsługi wydajnością. W związku z powyższym warto zastosować sprawdzone rozwiązania przedstawione w specyfikacja Java Enterprise Edition (JEE). Do JEE zaliczają się lub współpracują między innymi, mechanizm wyszukiwania usług JNDI lub metoda opisu stron WWW w sposób skryptowy za pomocą Java Server Pages (JSP).

Celem usprawnienia procesu tworzenia niniejszego oprogramowania zastosowano mechanizmy pomagające w zarządzaniu przebiegiem obsługi aplikacji, do czego została użyta funkcjonalność biblioteki Struts2. Mechanizmy umożliwiające budowanie stron interfejsu użytkownika na podstawie wcześniej zdefiniowanego wzorca wspomaga wykorzystanie biblioteki Tiles2.

Podstawową jednostką organizacyjną w programie wykorzystującym szkielet programistyczny Struts2 jest akcja. W najprostszej konfiguracji akcja jest jednostką identyfikującą się na poziomie konfiguracji nazwą i jednym plikiem jsp, który zapewni efekt wizualny jej wykonania. Adres strony internetowej zdefiniowanej „akcja” jest tworzony na podstawie jej nazwy. Domyślnie do nazwy akcji jest dołączany napis „.action”, jednakże można to łatwo zmienić w pliku konfiguracyjnym aplikacji. Przykładowa definicja prostej akcji wygląda następująco:

```
<action name=”HelloWorld”>
          <result>/HelloWorld.jsp</result>
</action>
```

Adres akcji będzie wglądał następująco: HelloWorld.action, a wynikiem wywołania strony o tym adresie będzie kod HTML zdefiniowany w pliku HelloWorld.jsp. Trzeba jednak zauważyć, że akcja zdefiniowana w tak ubogi sposób nie zapewnia praktycznie żadnej funkcjonalności, która mogłaby zostać wykorzystana do budowy większej aplikacji.

W aplikacji opartej o Struts2 obiekty akcji są poddawane działaniu interceptorów. Interceptory wykonują preprocessing, czyli przygotowują akcję do wykonania działania i postprocessing, w którym wykorzystują wyniki wygenerowane przez akcję. Interceptory są układane w stosy, których wykorzystanie może definiować każda akcja z osobna. W prototypie serwera opracowanego w ramach niniejszej pracy przewidziano dwa stosy interceptorów, z czego jeden jest przeznaczony do obsługi akcji

wymagających zalogowanego użytkownika, a drugi obsługuje akcje dostępne dla każdego.

Stos odpowiedzialny za obsługę akcji przeznaczonych wyłącznie dla zalogowanego użytkownika charakteryzuje się występowaniem interceptora login. Interceptor ten jest zdefiniowany następująco:

```
<interceptor name="login"
class="pl.toskl.webapp.struts2.LoginInterceptor" />
```

a stos, w którym jest wykorzystany następująco:

```
<interceptor-stack name="loginStack">
                <interceptor-ref name="servlet-config" />
                <interceptor-ref name="exception"/>
                <interceptor-ref name="alias"/>
                <interceptor-ref name="params"/>
                <interceptor-ref name="prepare" />
                <interceptor-ref name="chain" />
                <interceptor-ref name="model-driven" />
                <interceptor-ref name="fileUpload" />
                <interceptor-ref name="static-params" />
                <interceptor-ref name="params" />
                <interceptor-ref name="validation">
                  <paramname="excludeMethods">
                    input,back,cancel,browse
                  </param>
                </interceptor-ref>
                <interceptor-ref name="conversionError"/>
                <interceptor-ref name="workflow">
                  <param name="excludeMethods">
                    input,back,cancel,browse
                  </param>
                </interceptor-ref>
                <interceptor-ref name="login" />
</interceptor-stack>
```

Ważnymi i wykorzystywanymi w tej aplikacji interceptorami są:

- Servlet-config – jego zadaniem jest zapewnienie dostępu akcji do kontekstu. Jeżeli klasa akcji rozszerza interfejs SessionAware, do obiektu akcji za pomocą implementacji tego interfejsu zostanie włożona sesja HTTP, co znacząco ułatwia pracę;
- Params – jego zadaniem jest przepisanie parametrów HTTP Request do odpowiednich pól obiektu akcji, oczywiście tylko wówczas, gdy takie istnieją;

- FileUpload – odpowiada za obsługę wgrywania pliku na serwer. Jego zadaniem jest obsługa pola formularza HTML <INPUT type=”FILE”>. Plik wskazywany przez tag <FILE> zostanie pobrany na serwer i umieszczony w polach akcji typu java.io.File;
- Validation – odpowiada za wykorzystanie danych zawartych w plikach XML (standardowo pliki o nazwie [nazwa akcji|nazwa klasy]-validate.xml) opisujących sposób sprawdzenia poprawności danych zawartych w formularzu HTML i przesyłanych do serwera. Interceptor wywołuje również metodę akcji validate(), jeżeli akcja taką posiada;
- Login – interceptor utworzony na potrzeby tego projektu. Wymusza zdefiniowanie w konfiguracji aplikacji globalnego wyniku dla akcji o nazwie ‘login’, określającego miejsce przekierowania sterowania w momencie wykrycia próby dostępu użytkownika niezalogowanego do akcji objętej działaniem tego interceptora. Jego zadaniem jest również wykorzystanie danych autoryzacyjnych użytkownika, wpisanych na ekranie logowania, do stwierdzenia autentyczności podawanych danych i zalogowania użytkownika. Zalogowanie użytkownika skutkuje odczytaniem danych o użytkowniku z bazy danych i zapisaniem ich w sesji HTTP, co wiąże kolejne wywołania stron z konkretnym użytkownikiem. Globalny wynik akcji jest definiowany następująco:

```
<global-results>
  <result name="login" type="redirect-action">LoginInput
  </result>
</global-results>
```

Klasa implementująca interceptor musi rozszerzać interfejs Interceptor, który obliguje ją do implementacji metody:

```
public String intercept(ActionInvocation invocation)
```

Metoda ta jest wywoływana przed wywołaniem jakiejkolwiek metody należącej do akcji.

W pliku konfiguracyjnym akcje są często zdefiniowane w postaci dwóch wpisów. Nazwa pierwszej z akcji zakończona jest poprzez Input. Zdefiniowana w ten sposób

akcja jest wywoływana tylko przy pierwszym odwiedzeniu strony przez użytkownika. Akcja ‘Input’ wywołuje metodę akcji input(), której to wywołanie nie powoduje (co zaznaczono w definicji stosu interceptorow) sprawdzania poprawności danych i wyświetlania błędów dla pustych jeszcze pól akcji. Takie działanie jest podyktowane faktem, iż interceptor validator nie jest uruchamiany w momencie, gdy jest wywoływana metoda akcji o nazwie input(). Druga z pary akcji, która nie jest zakończona przez tekst Input, wywołuje z reguły metodę execute() odpowiedzialną za właściwe działanie akcji wraz z wywołaniem sprawdzania poprawności danych. Przyjęto taki sposób rozwiązania problemu w działaniu, jakim charakteryzuje się interceptor validator, aby komunikaty mówiące o braku poprawności wprowadzonych danych pojawiały się na ekranie w momencie wystąpienia rzeczywistego błędu poprawności, czyli nie przy pierwszym wejściu na konkretną stronę.

Zdefiniowane akcje z opisem ich funkcjonalności przedstawiono w poniższej tabeli (*Tabela 2*). W poniższej tabeli nie są wymienione akcje odpowiedzialne za inicjalizację stron (w nazwie akcji Input), gdyż nie służą rozwiązaniu zagadnienia stawianego przed tym oprogramowaniem i są narzutem technologii.

Tabela 2 Opis akcji/funkcji aplikacji serwerowej

Opis akcji/funkcji aplikacji serwerowej		
L.p.	**Opis funkcjonalności akcji**	**Definicja akcji**

1.	Strona umożliwia wpisanie danych użytkownika, takich jak nazwa użytkownika i hasło, które umożliwiają rozpoznanie użytkownika. Dane są wykorzystywane przez LoginInterceptor. Jeżeli dane są poprawne i użytkownik zostanie rozpoznany, jego dane są przechowywane w sesji HTTP oraz wykorzystywane w wielu miejscach aplikacji. Po poprawnym zalogowaniu zostanie wyświetlona strona zawierająca menu aplikacji.	`<action name="Login" class="web.Login">`
		` <result name="input">/site/tiles/Logowanie_t.jsp</result>`
		` <result type="redirect-action">Menu</result>`
		`</action>`

2.	Strona przedstawia menu aplikacji. Umożliwia przejście do stron przedstawiających:
	▪ Listę użytkowników;
	▪ Listę formularzy;
	▪ Listę wydruków;
	▪ Import danych.
		`<action name="Menu" method="execute" class="web.Menu">`
		` <result name="input">/site/tiles/Menu_t.jsp</result>`
		`</action>`

3.	Wyświetla listę użytkowników występujących w aplikacji.	`<action name="UzytkownicyInput" method="input"`
		`class="web.Uzytkownicy">`
		` <result name="input">/site/tiles/Uzytkownicy_t.jsp</result>`
		`</action>`

4.	Wyświetla stronę z danymi wybranego użytkownika umożliwia ich modyfikację:
	▪ Nazwa użytkownika;
	▪ Hasło;
	▪ Imię i nazwisko;
		`<action name="UserEdit" method="execute" class="web.UserEdit">`
		` <result name="input">/site/tiles/UserEdit_t.jsp</result>`
		` <result type="redirect-action">UzytkownicyInput</result>`
		`</action>`

	▪ Aktywny/nieaktywny.	
5.	Wyświetla stronę zawierającą dane wybranego użytkownika z pytaniem o potwierdzenie wyłączenia dostępu.	`<action name="UserDelete" method="execute" class="web.UserDelete"> <result type="redirect-action">UzytkownicyInput</result> </action>`
6.	Umożliwia wpisanie danych nowego użytkownika.	`<action name="UserNew" method="execute" class="web.UserNew"> <result name="input">/site/tiles/UserNew_t.jsp</result> <result type="redirect-action">UzytkownicyInput</result> </action>`
7.	Wyświetla listę formularzy dostępnych w aplikacji. Umożliwia podjęcie następujących akcji: ▪ Dodanie nowego formularza; ▪ Edycja danych wybranego formularza; ▪ Pobranie pliku zawierającego opis formularza; ▪ Pobranie pliku tła wybranego formularza.	`<action name="FormularzeInput" method="input" class="web.Formularze"> <result name="input">/site/tiles/Formularze_t.jsp</result> </action>`
8.	Wyświetla dane wybranego formularza oraz umożliwia edycję: ▪ Nazwy formularza; ▪ Pliku tła; ▪ Aktywny/nieaktywny.	`<action name="FormularzEdit" method="execute" class="web.FormularzEdit"> <result name="input">/site/tiles/FormularzEdit_t.jsp</result> <result type="redirect-action">FormularzeInput</result> </action>`
9.	Umożliwia import pliku zawierającego opis formularza przygotowanego w zewnętrznym programie Edytor Formularzy.	`<action name="FormularzNew" method="execute" class="web.FormularzNew"> <result name="input">/site/tiles/FormularzNew_t.jsp</result> <result type="redirect-action">FormularzAdditionInput</result> </action>`
10.	Po odczytaniu zaimportowanego pliku opisującego formularz umożliwia wprowadzenie danych przydatnych w pracy z formularzami, takich jak:	`<action name="FormularzAddition" method="execute" class="web.FormularzAddition"> <result`

	▪ Nazwa formularza; ▪ Plik tła; ▪ Aktywny/nieaktywny. Jeżeli plik opisujący formularz zawiera ww. dane, to są one podpowiadane.	name="input">/site/tiles/FormularzAddition_t.jsp</result> <result type="redirect-action">FormularzeInput</result> </action>
11.	Wywołanie tej strony powinno zawierać parametr będący unikalnym identyfikatorem formularza. Na podstawie identyfikatora strona umożliwia pobranie pliku zawierającego opis formularza.	<action name="FormularzDownload" method="execute" class="web.Download"> <result name="success" type="stream"> <param name="contentType">text/xml</param> <param name="inputName">inputStream</param> <param name="contentDisposition">attachment;filename="${filename}"</param> <param name="bufferSize">1024</param> </result> </action>
12.	Wywołanie tej strony powinno zawierać parametr będący unikalnym identyfikatorem formularza. Na podstawie identyfikatora strona umożliwia pobranie pliku tła formularza.	<action name="FormularzBackgroundDownload" method="executeback" class="web.Download"> <interceptor-ref name="notLoginStack"/> <result name="success" type="stream"> <param name="contentType">image</param> <param name="inputName">inputStream</param> <param name="contentDisposition">filename="${filename}"</param> <param name="bufferSize">1024</param> </result> </action>
13.	Umożliwia pobranie danych konkretnego wydruku.	<action name="DaneDownload" method="executedane" class="web.Download"> <result name="success" type="stream"> <param name="contentType">text/xml</param> <param name="inputName">inputStream</param> <param

		name="contentDisposition">attachment;filename="${filename}"</param> <param name="bufferSize">1024</param> </result> </action>
14.	Import danych wydruku. Na podstawie danych zawartych w pliku rozpoznawany jest formularz, do którego przynależą dane. W przypadku niezgodności danych z formularzem lub nieodnalezienia odpowiedniego formularza, zostanie zgłoszony błąd, a import nie dojdzie do skutku.	<action name="ImportData" method="execute" class="web.ImportData"> <result name="input">/site/tiles/ImportData_t.jsp</result> <result type="redirect-action">Menu</result> </action>
15.	Wyświetla listę zaimportowanych danych. Lista ta jest listą dostępnych/wykonanych wydruków. Umożliwia pobranie: ▪ Danych w pliku tekstowym; ▪ Wydruku w postaci pliku PDF z tłem; ▪ Wydruku w postaci pliku PDF bez tła.	<action name="DaneView" method="input" class="web.DaneView"> <result name="input">/site/tiles/DaneView_t.jsp</result> </action>
16.	Umożliwia wygenerowanie oraz pobranie pliku PDF zawierającego wydruk z tłem.	<action name="PrintWithBackground" method="printwithbackground" class="web.Print"> <result name="success" type="stream"> <param name="contentType">application/pdf</param> <param name="inputName">inputStream</param> <param name="contentDisposition">filename="${filename}"</param> <param name="bufferSize">1024</param> </result> </action>
17.	Umożliwia wygenerowanie oraz pobranie pliku PDF zawierającego wydruk bez tła.	<action name="PrintWithoutBackground" method="printwithoutbackground" class="web.Print"> <result name="success" type="stream"> <param name="contentType">application/pdf</param>

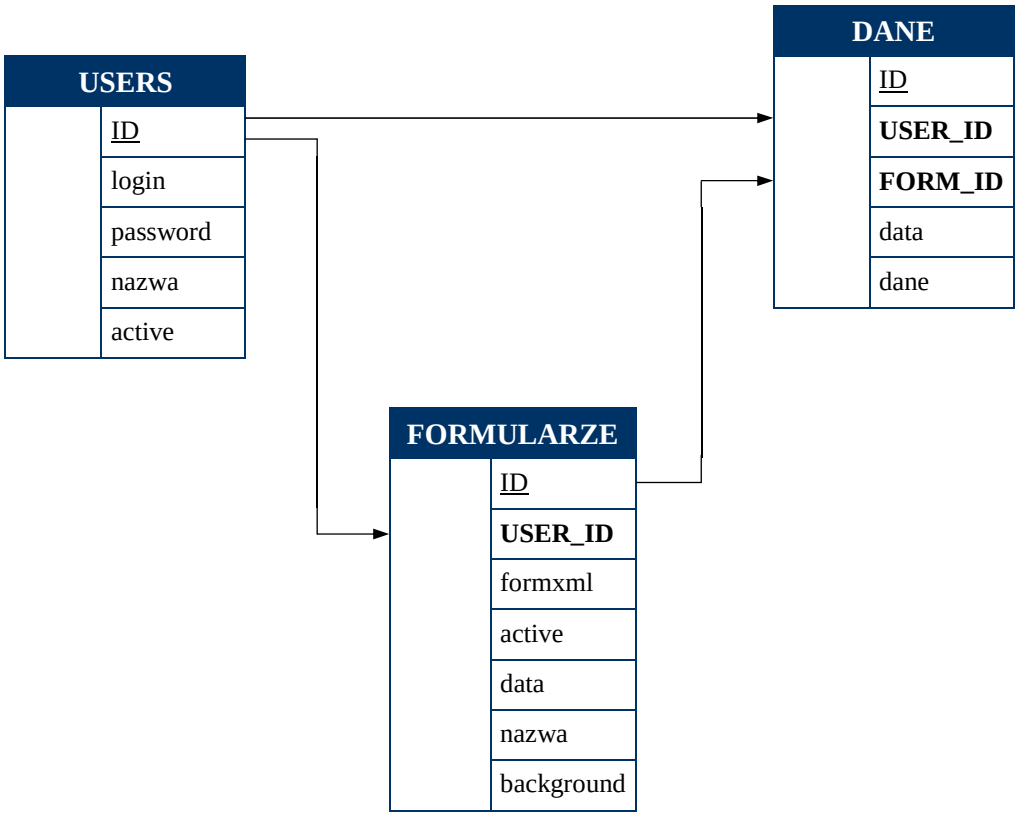
		<param name="inputName">inputStream</param> <param name="contentDisposition">filename="${filename}"</param> <param name="bufferSize">1024</param> </result> </action>
18.	Strona udostępnia funkcjonalność niezbędną do importu danych wydruku z zewnętrznego programu. Strona ta jest dostępna dla niezalogowanych użytkowników, jednak wymaga podania niezbędnych danych identyfikujących użytkownika jako parametrów wywołania. Nie posiada reprezentacji graficznej. Parametrami wywołania są: ▪ Nazwa użytkownika; ▪ Hasło użytkownika; ▪ Dane do wydruku wypełniające formularz. Jeżeli stronę wywołuje niezalogowany użytkownik (zewnętrzny program), wynikiem akcji jest strona zawierająca identyfikator zaimportowanych danych niezbędny do pobrania wydruków. Jeżeli stronę wywołuje zalogowany użytkownik (posiadający w sesji HTTP dane zalogowanego użytkownika), wyświetlana jest strona umożliwiająca pobranie wydruków w postaci plików PDF z tłem lub bez tła. Z tego ostatniego sposobu korzysta element aplikacji serwera stworzony w Google Web Toolkit, działający w przeglądarce na prawach i z dostępem do sesji zalogowanego użytkownika.	<action name="Wrzutnia" method="execute" class="web.Wrzutnia"> <interceptor-ref name="notLoginStack"/> <result>/site/Wrzutnia.jsp</result> <result name="getpdf">/site/tiles/WrzutniaOnline_t.jsp</result> </action>
19.	Umożliwia pobranie przez niezalogowanego użytkownika wydruków w pliku PDF z tłem. Parametry wywołania muszą posiadać informacje o nazwie użytkownika i haśle umożliwiające identyfikację użytkownika.	<action name="GetPdfWithBackground" method="printwithbackground" class="web.Wrzutnia"> <interceptor-ref name="notLoginStack"/> <result name="success" type="stream"> <param name="contentType">application/pdf</param> <param name="inputName">inputStream</param>

		<param name="contentDisposition">filename="${filename}"</param> <param name="bufferSize">1024</param> </result> <result name="notLogin">/site/Wrzutnia.jsp</result> </action>
20.	Umożliwia pobranie przez niezalogowanego użytkownika wydruków w pliku PDF bez tła. Parametry wywołania muszą posiadać informacje o nazwie użytkownika i haśle umożliwiające identyfikację użytkownika.	<action name="GetPdfWithoutBackground" method="printwithoutbackground" class="web.Wrzutnia"> <interceptor-ref name="notLoginStack"/> <result name="success" type="stream"> <param name="contentType">application/pdf</param> <param name="inputName">inputStream</param> <param name="contentDisposition">filename="${filename}"</param> <param name="bufferSize">1024</param> </result> <result name="notLogin">/site/Wrzutnia.jsp</result> </action>
21.	Ta strona umożliwia wybranie z listy formularza, który będzie wypełniany w trybie online.	<action name="FillFormOnlineStart" class="web.FillFormOnlineStart"> <result>/site/tiles/FillFormOnlineStart_t.jsp</result> </action>
22.	Wybrany formularz jest wykorzystywany przez element programu stworzony za pomocą Google Web Toolkit do wypełniania formularzy w trybie online.	<action name="FillFormOnline" class="web.FillFormOnline"> <result>/site/tiles/FillFormOnline_t.jsp</result> </action>

Źródło: Opracowanie własne

6.5. BAZA DANYCH

Baza danych oparta o serwer PostgreSql (patrz [6]) zbiera dane wprowadzane za pomocą opracowywanego oprogramowania. Dane te umożliwiają identyfikację i autoryzację użytkowników, zapamiętywanie przygotowanych szablonów formularzy oraz danych wypełniających formularze. Struktura bazy zapewnia spójność, a zastosowanie PostgreSql - trwałość i transakcyjność. Model danych przechowywanych w bazie danych został zaprezentowany na poniższych schemacie (*Schemat 9*).



Schemat 9 Schemat bazy danych

Źródło: Opracowanie własne

6.6. WYPEŁNIANIE FORMULARZY ONLINE

Oprogramowanie wykonywane w ramach niniejszej pracy byłoby niepełne bez możliwości wypełniania utworzonych formularzy bezpośrednio w przeglądarce.

Zapewnienie wygodnego korzystania z formularzy implikuje zapotrzebowanie na skomplikowany interfejs, którego zachowanie i sposób obsługi powinny naśladować znane przez użytkownika produkcje. Do budowy interfejsu opisującego formularze w przeglądarce internetowej w przystępnej formie konieczne jest użycie JavaScript, który zapewni dynamikę i prostotę obsługi. Celem zapewnienia łatwego procesu tworzenia i utrzymania tego elementu aplikacji, wykorzystano narzędzie Google Web Toolkit (GWT). Narzędzie to umożliwia tworzenie aplikacji w sposób znany programiście języka Java, z wykorzystaniem bibliotek Swing i Awt. Obsługa zdarzeń wywoływanych przez użytkownika, zachodzących w aplikacji za pomocą mechanizmu Listenerów jest identyczna w obu przypadkach. GWT przyjmuje kod programu w języku Java i tłumaczy go na język JavaScript, który jest obsługiwany i wykonywany przez wszystkie znaczące przeglądarki internetowe. GWT dodatkowo zapewnia jednorodne zachowanie wygenerowanego kodu w każdej przeglądarce, pomimo różnej interpretacji przez nie poleceń języka.

Program ten korzysta ze zmodyfikowanego modelu danych opracowanego dla aplikacji, gdyż GWT nie obsługuje mechanizmu PropertyChangeListener, jak również nie posiada wbudowanych klas Point, Rectangle w oryginalnym modelu danych pochodzących od klas z biblioteki AWT.

Wywołanie programu przewiduje trzy parametry:

- Adres na który zostaną odesłane zebrane dane;
- Adres z którego pobrać zdjęcie tła;
- Opis formularz stworzony w edytorze formularzy w formie tekstowej.

Plik HTML odpowiedzialny za uruchomienie wypełniania formularzy online w którym zaznaczono kursywą znaczące parametry (w środkowej części ciała strony) wygląda następująco:

```
<html>
<head>
<META HTTP-EQUIV="Content-type" CONTENT="text/html; charset=UTF-8"/>
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<link rel="StyleSheet" href="/FormularzeServer/site/style/style.css"
type="text/css" media="screen" />
```

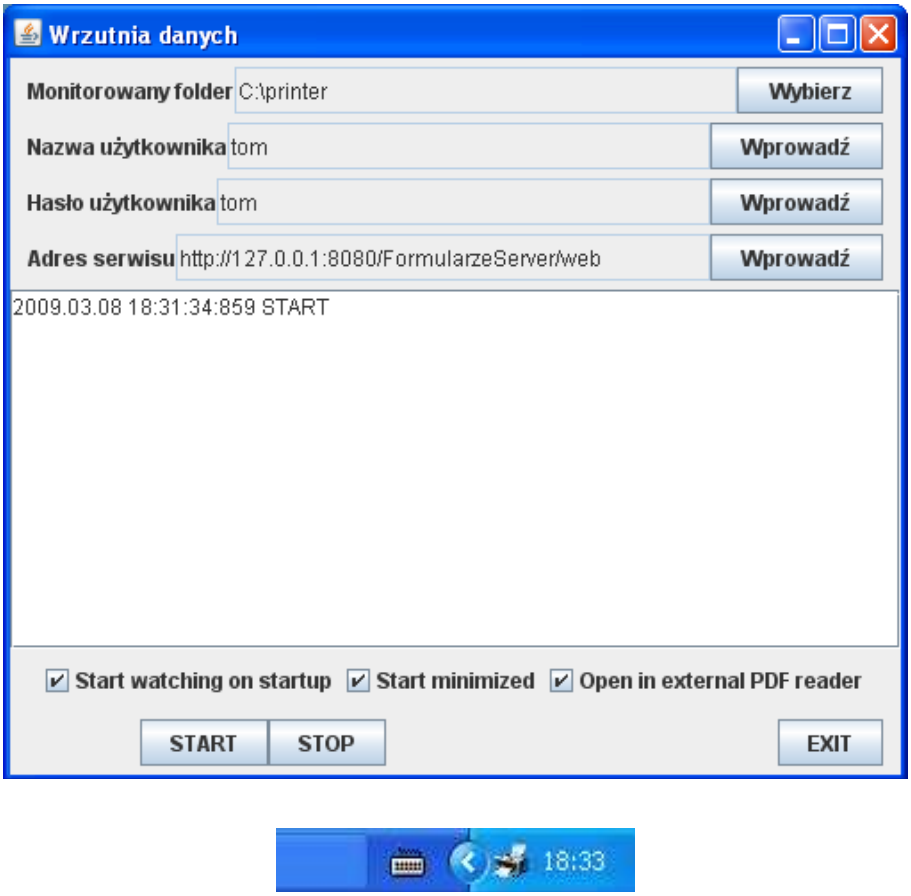
```
<title>Wprowadzanie danych - Online</title>
</head>

<body>
<script type="text/javascript" language="javascript"
src="/FormularzeServer/site/js/pl.toskl.formularze.gwt.BaseCanvas.noca
che.js"></script>

<input type="hidden" id="submitAction"
value="/FormularzeServer/web/Wrzutnia.action"/>
<input type="hidden" id="image"
value="/FormularzeServer/web/FormularzBackgroundDownload.action?id=12"
/>
<input type="hidden" name="formularz" value="
<FORM>
<NAME>Urlop bezplatny</NAME>
(...)
</FORM>
" id="formularz"/>

<center>
<span id="canvas"></span>
<span id="zapisz"></span>
</center>
</body>
</html>
```

6.7. PROGRAM EKSPORTUJĄCY DANE

Działający na serwerze program generujący wydruki i edytor formularzy udostępniają obsługę wydruków, która byłaby niepełna bez prezentowanego programu do eksportu danych. Umożliwia on korzystanie z funkcji udostępnianych przez serwer w sposób niewidoczny dla użytkownika. Ponadto, dzięki jego zastosowaniu, wszystkie dane opuszczające firmę w postaci papierowych wydruków są zmagazynowane na serwerze w formie elektronicznej. Podstawowym zadaniem programu jest przesłanie danych wypełniających wydruk do serwera aplikacji i pobranie wydruku, przy czym dane są przygotowywane przez zewnętrzny program. Dane muszą zostać przygotowane w odpowiednim formacie, zgodnym z formatem formularza zachowanym w aplikacji serwerowej, tak aby możliwe było połączenie danych z formularzem i przygotowanie wydruku.

Program eksportujący dane został napisany w języku Java z wykorzystaniem bibliotek dostępnych w wersji JRE 1.6. Parametryzacja programu umożliwia mu nawiązanie połączenia z serwerem w określonej lokalizacji i autoryzację użytkownika. Do

połączenia wykorzystywany jest protokół HTTP, który został zaimplementowany w bibliotece Apache HTTPClient, udostępnianej przez organizację Apache Software Foundation.



Rysunek 9 Interfejs Programu Eksportującego Dane

Źródło: Print Screen z Programu do eksportu danych

Interfejs użytkownika został utworzony z wykorzystaniem biblioteki Swing i Awt dostarczanej z Java Runtime Environment w wersji 1.6. Wspomniana wersja tych bibliotek udostępnia klasę TaskIcon, za pomocą której można budować aplikacje reprezentowane poprzez ikonę w obszarze obok zegara systemowego na pasku Windows, bez wykorzystywania zewnętrznych bibliotek.

Na obrazku (**Rysunek 9**) został przedstawiony wygląd programu uruchomionego pod kontrolą systemu Windows. Interfejs użytkownika umożliwia zdefiniowanie:

- Monitorowanego katalogu na dysku;
- Nazwy użytkownika;
- Hasła użytkownika;
- Lokalizacji aplikacji serwera.

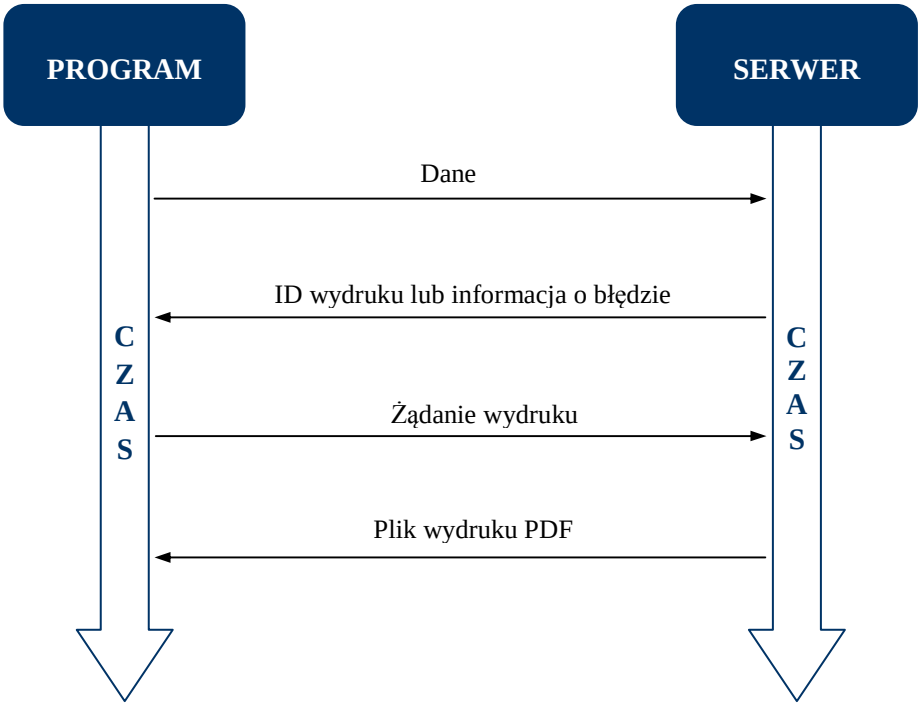
Prezentowane są również przełączniki określające sposób działania aplikacji:

- Załączenie monitorowania przy starcie programu;
- Sposób prezentacji programu po uruchomieniu, start zminimalizowanego programu do Tray;
- Wyświetlenie wydruku po odebraniu pliku pdf.

Ponadto interfejs przedstawia pole, którego zadaniem jest informowanie użytkownika o aktualnym stanie programu, jak również o historii wykonanych operacji. Pole to pełni funkcję ulotnego, niezapisywanego do pliku logu.

Dla ułatwienia obsługi i odciążenia użytkownika z potrzeby wprowadzania parametrów za każdym uruchomieniem aplikacji, parametry są przechowywane w postaci pliku XML obsługiwanego przez klasę z biblioteki standardowej Properties, która dba o jego poprawny zapis i odczyt.

Komunikacja z serwerem została przedstawiona na poniższym schemacie (*Schemat 10*).



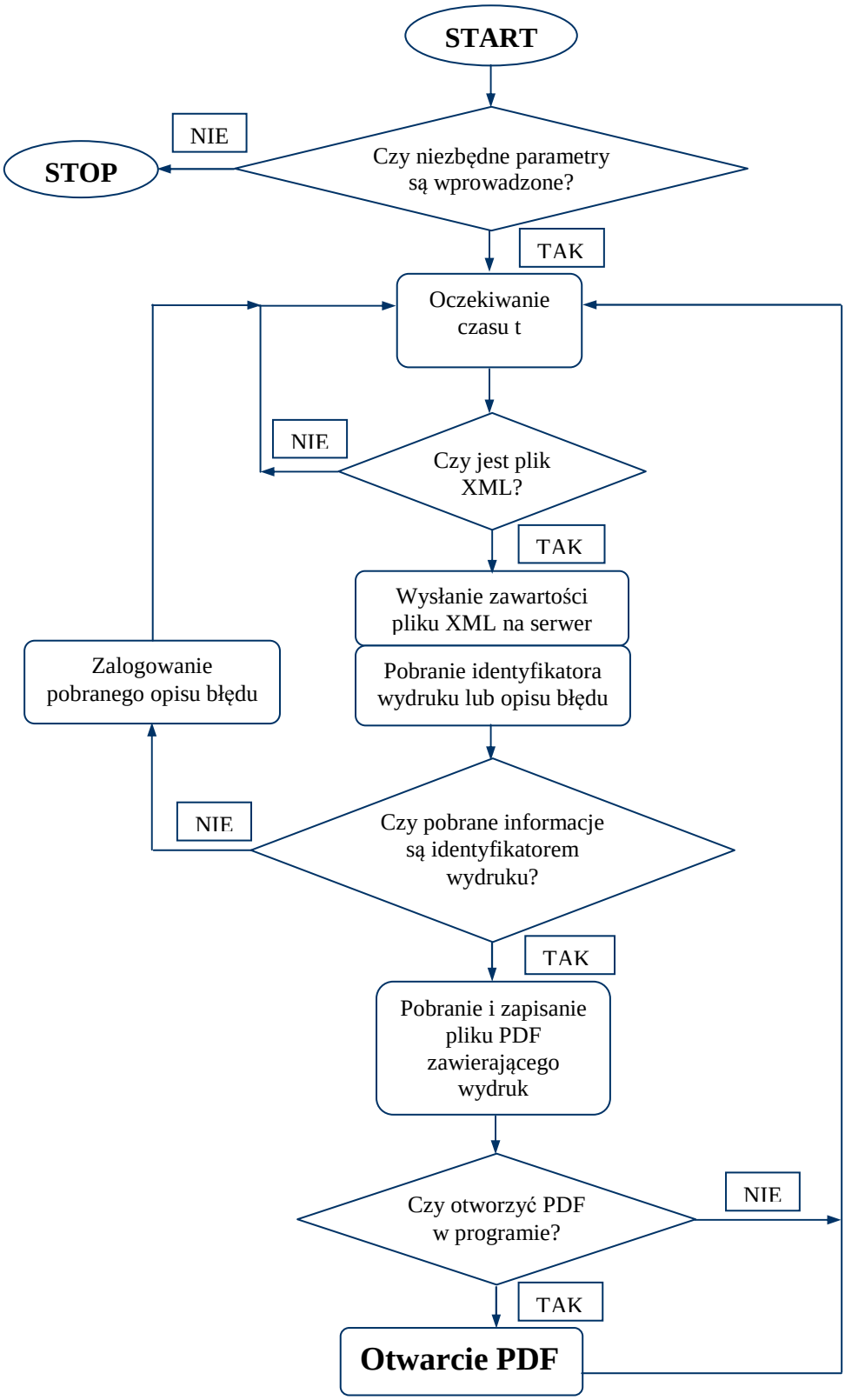
Schemat 10 Schemat działania programu eksport

Źródło: Opracowanie własne

Po wybraniu monitorowanego katalogu i uruchomieniu aplikacji przyciskiem START, zostanie rozpoczęte sprawdzanie czy wybrany katalog zawiera pliki XML. Jeżeli w podanym katalogu zostanie odnaleziony plik XML, będzie to oznaczało rozpoczęcie procedury wysyłania tego pliku do serwera. Wysłane dane, po poprawnej autoryzacji, zostaną zapisane na serwerze oraz zostanie odesłany identyfikator wydruku. Na podstawie identyfikatora aplikacja pobierze przygotowany przez serwer plik PDF, który zostanie zapisany w monitorowanym katalogu. W zależności od konfiguracji, pobrany plik zostanie wyświetlony w programie określonym w systemie operacyjnym jako domyślny, służący do otwierania plików w formacie PDF.

Nazwa użytkownika i hasło są wymagane w procesie autoryzacji użytkownika, co jest niezbędne do przyjęcia danych przez serwer. Informacje o nazwie użytkownika i haśle są wysyłane wspólnie z danymi odczytanymi z pliku danych wydruku.

Schemat blokowy działania programu wygląda następująco *(Schemat 11*):



Schemat 11 Schemat blokowy działania Programu Eksportującego Dane

Źródło: Opracowanie własne

7. ZALETY, WADY ORAZ PLANY ROZWOJOWE

Prototypy opracowane na potrzeby tej pracy spełniają większość z postawionych przed nimi we wstępie do pracy wymagań. Jednak produkt ten nie może być uznany za skończony, pełny czy w formie ostatecznej, gotowej do sprzedaży. Jeżeli produkt miałby zostać wprowadzony na rynek, jego funkcjonalność musiałaby zostać mocno rozbudowana, aby doścignąć funkcjonalność produktów istniejących na rynku, a tym samym poprawić jego konkurencyjność. Jednakże w pewnych aspektach prototyp podchodzi do problemu w nowatorski, wcześniej niespotykany sposób.

Prototypy będące wynikiem tej pracy zostały opracowane z wykorzystaniem języka i technologii Java. Ich budowa oraz zastosowanie mechanizmów umożliwiających uruchamianie aplikacji niezależnie od systemu operacyjnego stwarzają możliwość poszerzenia grona potencjalnych odbiorców. Jednym z ważniejszych aspektów jest także zastosowanie JVM (Java Virtual Machine), co skutkuje możliwością opracowania kodu aplikacji jak również poprawek, rozszerzeń dla różnych systemów operacyjnych jednocześnie.

Elementem, który w powstałych prototypach zasługuje na uznanie za zaletę, jest takie opracowanie modelu danych, że jego wykorzystanie we wszystkich aplikacjach jest możliwe praktycznie bez zmian.

7.1. EDYTOR FORMULARZY, PROGRAM WYSYŁAJĄCY DANE NA SERWER

Wykorzystanie w tych aplikacjach możliwości oferowanych przez standardową bibliotekę Swing i AWT skutkuje dobrym wyglądem oraz zachowaniem aplikacji, zgodnie z oczekiwaniami użytkownika. Jednocześnie wykorzystanie modelu komponentowego w opracowanym Edytorze Formularzy daje możliwość zamknięcia w spójnym obiekcie wszystkich funkcjonalności, które były potrzebne do wyrysowywania formularza. Jeżeli nastąpi potrzeba przebudowania działania aplikacji lub wykorzystania jej elementów w innym miejscu, nie będzie stanowiło to problemu; co więcej, będzie wykonalne bez ingerencji w znaczną cześć programu.

Wykorzystanie mechanizmów dostarczanych w standardowej bibliotece udostępnianej wraz z instalacją środowiska uruchomieniowego Java daje pewność, iż program na różnie skonfigurowanych komputerach będzie zachowywał się identycznie. Jednocześnie zarządzanie wyglądem za pomocą LayoutManagerów pozwoli w przyszłości na łatwą zmianę wyglądu aplikacji.

Aby aplikacja Edytor Formularzy mogła zostać wprowadzona na rynek i stać się konkurencją dla istniejących rozwiązań, łatwo wyobrazić sobie funkcjonalności, o które musiałaby zostać rozbudowana. Jedną z nich jest możliwość ustalania parametrów czcionki w konkretnym polu, takich jak wielkość, krój, kolor. Dzięki temu można by uzyskać zróżnicowanie wyglądu danych pojawiających się na wydruku. Wczytywany plik tła w tej wersji programu musi zostać przygotowany w zewnętrznym programie graficznym. Prototyp nie umożliwia przeskalowania zdjęcia tła lub obcięcia zbędnych części np. marginesów. Bardzo przydatną funkcją byłaby również możliwość przybliżania i oddalania widoku obszaru roboczego, co ułatwiałoby precyzyjne zaznaczanie pól danych.

Kolejnym elementem rozwijającym prototyp byłoby wbudowanie w aplikacje prostego języka skryptowego, za pomocą którego można by przykładowo: dodawać ze sobą dwie wartości, obliczać procenty, itp. Wprowadzenie takiego rozwiązania niesie za sobą potrzebę określenia jakiego typu dane przedstawia dane pole, jak również umożliwienie zgłaszania niepoprawności danych względem typu, jaki dana komórka powinna

przyjmować. Wprowadzanie i sprawdzenie poprawności kodu odbywałoby się w Edytorze Formularzy, natomiast jego wykonanie stałoby się elementem serwera..

7.2. APLIKACJA SERWEROWA

Oprogramowanie serwerowe będące częścią tej pracy działa w środowisku Tomcat i wykorzystuje część funkcjonalności wynikającej ze specyfikacji J2EE. Głównie stosowane są możliwości takie jak JSP, serwlety oraz JNDI. Nadal pozostaje jednak część funkcji specyfikacji J2EE tutaj niewykorzystywanych, a których zaimplementowanie mogłoby przyczynić się do poprawy wybranych charakterystyk omawianej aplikacji. Istnieje wiele możliwości rozbudowy aplikacji z zastosowaniem specyfikacji J2EE. Przykładowo, zastosowanie EJB (Enterprise Java Beans) oraz JMS (JavaMessage Server) mogłoby skutkować zwiększeniem wydajności i modularyzacji omawianej aplikacji, co w efekcie umożliwi podział wykonywanych zadań pomiędzy wiele maszyn.

Jeżeli serwer, na którym działa omawiane oprogramowanie, zostałby mocno obciążony, prawdopodobne jest, iż wystąpiłby problemy z dostępnością usług. W związku z tym, iż Tomcat pochodzi z rodziny Apache, możliwe jest uruchomienie oddzielnego serwera HTTP Apache (z możliwością uruchomienia szyfrowania połączeń), który przejmie prace związane z nawiązywaniem i utrzymywaniem połączeń. Innym dostępnym rozwiązaniem w serwerze HTTP Apache jest zastosowanie balansowania obciążenia (z ang. *load balancing*) do kilku maszyn obsługujących kontener Tomcat.

W omawianej aplikacji dostęp do bazy danych jest zapewniony przez środowisko kontenera Tomcat. Kontener ten zarządza pulą połączeń, które w odpowiednim momencie mogą zostać wykorzystane przez aplikację. Zastosowanie puli połączeń zapewnia zarządzanie liczbą połączeń oraz służy głównie zapewnieniu stabilności bazy danych, jak również sensownego czasu odpowiedzi na zapytania (w tym mniejszą liczbę zależnych od siebie jednoczesnych transakcji – zmniejszenie liczy zakleszczeń).

Zastosowano również JNDI (Java Naming and Directory Interface), co daje kolejny poziom konfiguracji i brak uzależnienia oprogramowania od konkretnego serwera bazodanowego. Dzięki temu użytkownik może uruchomić oprogramowanie

np. z wykorzystaniem działającego w firmie serwera bazy danych, do którego posiada już komercyjną licencję.

Niewątpliwą zaletą utworzonego prototypu jest zastosowanie w nim szkieletu programistycznego (z ang. *framework*) Struts2, którego główne zadania obejmują udostępnianie programowi za pomocą interceptorów danych nienależących bezpośrednio do programu (np. Sesja HTTP), jak również zebranie w jednym czytelnym pliku konfiguracji akcji możliwych do wykonania przez aplikację. Definicja akcji zawiera również definicję wyników danej akcji, dzięki czemu możliwe jest w łatwy sposób oprogramowanie i śledzenie przebiegu wywołań programu.

Dużym ułatwieniem dla programisty rozwijającego ten prototyp będzie zastosowanie biblioteki Tiles2, której działanie uwidacznia się w warstwie widoku. Rozszerzenie standardowego działania JSP o Tiles2 daje możliwość łączenia plików JSP w schemacie strony zawartym w pliku wzorca Tiles2. Modułowa konfiguracja strony ułatwia wprowadzanie zmian w jednym miejscu i ich propagację na wszystkie strony zawierające zmieniany kod. Wykorzystanie modułowej budowy stron www skutkuje również mniejszą ilością pracy, a także zmniejszeniem ilości powtarzającego się kodu.

7.3. WYPEŁNIANIE FORMULARZY ONLINE

Odnośnie rozbudowanego interfejsu usprawniającego pracę nad wypełnianiem formularzy bezpośrednio w przeglądarce internetowej, w tym momencie ciężko jest zaklasyfikować ten element jako wadę bądź zaletę. Założenia oparte na wykorzystaniu Google Web Toolkit, a co za tym idzie wyprodukowanie kodu JavaScript działającego we wszystkich przeglądarkach internetowych w ten sam sposób na podstawie kodu Java, może być jak najbardziej poczytane jako zaleta. Jednak GWT nie oferuje w swojej bibliotece obiektu Panelu (kontener komponentów) funkcjonalnie połączonego z obiektem Canvas (rysowanie), na którym można by wyświetlić zdjęcie (tło), a następnie w dowolnym miejscu umieścić pola edycyjne, które zostaną wypełnione przez użytkownika danymi. Tą funkcjonalność daje obiekt klasy wyprodukowanej przez niezależnego programistę, która będzie wchodziła w skład biblioteki GWT w kolejnych jego wersjach. W tym momencie wykorzystany komponent działa poprawnie tylko

w przeglądarce Internet Explorer firmy Microsoft, co jest zachowaniem niezgodnym z założeniami i należy to traktować jako wadę produktu.

8. PODSUMOWANIE

Wykorzystanie rozbudowanych aplikacji sieciowych działających w przedsiębiorstwach jest konieczne, aby mogły one zachować rentowność, jak również zdobyć przewagę nad konkurencją. Odpowiednie oprogramowanie ułatwia, a nawet umożliwia zarządzanie przedsiębiorstwem, zwłaszcza takim, które jest rozproszone geograficznie. Przykładowy problem budowy oprogramowania przestawiony w tej pracy nakłada na nie konkretne założenia. Dobrze zaprojektowana aplikacja powinna, po pierwsze, spełniać założenia biznesowe i przewidywać możliwość ich ewoluowania. Po drugie, powinna być przyjazna w odbiorze, prezentować czytelny oraz prosty w obsłudze interfejs, tak aby osoba, która nie brała udziału w procesie tworzenia oprogramowania, nie miała problemów z jej obsługą.

W ramach niniejszej pracy przedstawiono zasady tworzenia oprogramowania dostępnego za pomocą Internetu oraz dokonano wyboru odpowiednich do zrealizowania tego zadania bibliotek programistycznych i technologii. Opracowany zestaw prototypów realizuje założenia biznesowe, które zdefiniowano jako „tworzenie i obsługa formularzy za pomocą www", w możliwe optymalny sposób. Przestawia on budowę aplikacji działającej za pomocą sieci komputerowej, w której wykorzystano biblioteki i techniki programistyczne wpływające na uniwersalność aplikacji. Odpowiednie ich zastosowanie daje możliwość rozbudowy oraz dostosowania do zastanych elementów infrastruktury, a także zmieniających się potrzeb potencjalnego klienta. Oprogramowanie wchodzące w skład zestawu prototypów zostało podzielone na oddzielne projekty w taki sposób, aby umożliwić w przyszłości bezproblemowe wprowadzanie zmian i poprawek. Każdy z projektów charakteryzuje się innym sposobem komunikacji z użytkownikiem. Program służący do tworzenia formularzy jest

programem graficznym pozwalającym na wyrysowanie za pomocą myszy miejsc właściwych dla danych. Program eksportujący dane do serwera aplikacji przez większość czasu powinien działać w sposób niewidoczny dla użytkownika. Aplikacja serwerowa posługuje się stronami internetowymi do komunikacji z użytkownikiem. Prototypy powstałe w ramach tej pracy mogą pracować pod różnymi systemami operacyjnymi, a ponadto aplikacja serwerowa posiada możliwości skalowania wydajności za pomocą różnych technik oraz technologii.

Przedstawione powyżej właściwości sprawiają, że oprogramowanie to może być wykorzystane w dużej liczbie przedsiębiorstw, co więcej, może być dostosowane do potrzeb każdego z nich.

9. BIBLIOGRAFIA

[1] Dokumentacja Java Enterprise Edition, *Java EE at a Glance*, http://java.sun.com/javaee/

[2] Laurie B., Laurie P., *Apache. Przewodnik encyklopedyczny*, O'Reilly -Helion, Gliwice 2003

[3] Dokumentacja biblioteki Struts2, *Apache Struts2 Documentation Home*, http://struts.apache.org/2.x/docs/home.html

[4] Dokumentacja biblioteki Apache HTTPClient, *HttpClient Home*, http://hc.apache.org/httpclient-3.x/

[5] Pilone D., Pitman N., *UML 2.0 Almanach*, O'Reilly - Helion, Gliwice 2007

[6] Dybikowski Z., *PostgreSQL*, Helion, Gliwice 2001

10. SPIS TABEL

11. SPIS SCHEMATÓW

12. SPIS RYSUNKÓW