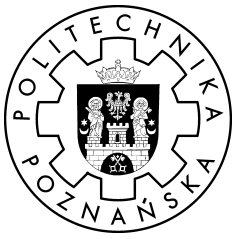
POLITECHNIKA POZNAŃSKA
WYDZIAŁ INFORMATYKI
INSTYTUT INFORMATYKI



INŻYNIERSKA PRACA DYPLOMOWA

System do zarządzania cyklem życia kart elektronicznych

SMART CARD MANAGEMENT SYSTEM

Paweł Brzeziński
Kamil Czyżnielewski
Dominik Minc
Maciej Sobkowiak

Promotor:
prof. dr hab. inż. Joanna Józefowska

Poznań, 2013

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie

Karty inteligentne od wielu lat stanowią podstawę bezpiecznej automatycznej identyfikacji. Ich rozmiar i kształt jest taki sam jak dla kart kredytowych. Za bezpieczeństwo danych odpowiada wbudowany, programowalny procesor. Karty inteligentne są o wiele bardziej popularne w Europie niż w Stanach Zjednoczonych. W Europie są one często używane w instytucjach zdrowotnych, czy w bankach. Dla przykładu każdy obywatel Niemiec posiada kartę inteligentną, którą posługuje się korzystając z opieki zdrowotnej [1].

Najczęstsze wykorzystanie kart inteligentnych:

- Systemy zabezpieczeń komputerowych
- Środek płatniczy w bankowości
- Bilety w komunikacji miejskiej
- Karty SIM w telefonii
- Karty stałego klienta i rabatowe

Mogą być one używane do kontroli dostępu do miejsc oraz sprzętu, jak również do rejestracji czasu pracy. Usługi jakie zapewniają karty inteligentne, to m. in. [4]:

- kontrola dostępu - zapobieganie dokonaniu nieuprawnionych operacji na zasobach pamięciowych karty (np. odczyt, zapis) - dostęp do zasobów możliwy jest po podaniu numeru PIN,
- uwierzytelnianie danych (np. za pomocą podpisu cyfrowego): potwierdzenie zgodności tożsamości źródła danych z deklarowaną, dostarczenie dowodu przez kartę, że jest źródłem wysłanych danych.
- uwierzytelnianie stron - potwierdzenie (jednostronne lub wzajemne) tożsamości stron uczestniczących w sesji komunikacyjnej - np. przy pomocy systemu kluczy (prywatnego i publicznego) i techniki challenge-response,
- usługa poufności - zapobieganie nieuprawnionemu ujawnieniu danych użytkownika podczas przesyłania i przechowywania danych (dane przechowywane są w postaci zaszyfrowanej, np. algorytmem DES),
- usługa integralności - zapobieganie nieuprawnionemu modyfikowaniu lub usunięciu

danych użytkownika podczas: przesyłania danych - przed wysłaniem danych wyznaczana jest wartość funkcji haszującej na tych danych, która następnie jest szyfrowana. Po otrzymaniu danych przez adresata następuje ponowne wyznaczenie wartości funkcji haszującej. Otrzymana wartość porównywana jest z dostarczoną przez nadawcę. Zgodność oznacza, że dane nie zostały zmienione podczas przesyłania. przechowywania.

- usługa niezaprzeczalności - zapobieganie odmowie potwierdzenia udziału w sesji komunikacyjnej przez stronę uczestniczącą w tej sesji.

W celu łatwego zarządzania tymi kartami, projektowane są systemy zarządzania kartami elektronicznymi. Jest to oprogramowanie umożliwiające zarządzanie kartami inteligentnymi przez cały cykl ich życia.

1.2 Opis problemu

Politechnika Poznańska używa osobnych systemów do zarządzania kartami pracowniczymi, studenckimi, czy też doktoranckimi. Wszystkie spełniają podobną rolę, dlatego też pojawił się pomysł na stworzenie jednego, spójnego systemu, który umożliwi zarządzanie różnymi typami kart.

Surowa karta dostarczana przez producenta nie nadaje się do bezpośredniego użytku w różnych zastosowaniach w życiu codziennym. W celu umożliwienia korzystanie z niej jako karty wstępu lub płatniczej należy przeprowadzić proces personalizacji. Jest to szereg czynności, który nadaje karcie unikatowe cechy, zwane profilem karty. Profil karty to zbiór aplikacji, które determinują charakter karty, jej nadruk graficzny oraz informacje zapisywane na jej chipie lub pasku magnetycznym.

Karta elektroniczna może zmieniać swój stan. Blankiety, które są nośnikami informacji mogą zostać uszkodzone. Aplikacje, takie jak np. bilety komunikacji miejskiej, muszą być czasowo odnawiane. Częstymi przypadkami są także kradzieże kart lub ich zagubienie. System zarządzający takimi kartami, musi więc posiadać obsługę wszystkich możliwych statusów karty. Szereg stanów, w których karta się znajduje, bądź historycznie znajdowała, nazywamy cyklem życia karty.

Pierwszym etapem cyklu życia karty jest zebranie danych o jej profilu oraz przygotowanie fizycznych nośników. Kolejny etap to proces personalizacji. Drukarki korzystają z wtyczek programowych, dzięki którym mogą pobierać zlecenia personalizacji. Wyposażone są także w czytniki, które rozpoznają model blankietu, na który ma nastąpić nadruk i wgranie danych. Dzięki zewnętrznemu programowi obsługującemu czytniki, wiadomo w jakich sektorach pamięci poszczególnych modelów blankietów zapisywać dane. Po pobraniu zlecenia personalizacji następuje nadruk danych na blankiet oraz instalacja aplikacji. Tak przygotowana karta jest spersonalizowana, czyli posiada odpowiednie indywidualne cechy i jest gotowa do użycia.

Rysunek 1.1: Cykl życia karty elektronicznej

1.3 Cel realizacji projektu

Głównym celem projektu było ułatwienie zarządzania kartami elektronicznymi w ciągu całego okresu jej użytkowania. Obejmowało to stworzenie aplikacji klienckiej, jak również centralnej części systemu.

Smart Card Management System stworzony został w celu zastąpienia istniejącej aplikacji odpowiadającej za zarządzanie legitymacjami pracowniczymi, będącej częścią systemu Kadry 2000. Ponadto jest on rozszerzony o dodatkowe funkcjonalności w stosunku do obecnie wykorzystywanego modułu oraz stanowi część już istniejącego systemu administracyjnego Politechniki Poznańskiej.

Zaprojektowany został z myślą o zarządzaniu różnymi typami kart. Dzięki temu za pomocą raz stworzonego oprogramowania możliwa będzie obsługa wielu klientów. Poprzez odseparowanie bazy danych, w łatwy sposób będzie można za pomocą tej samej aplikacji, obsługiwać zlecenia wewnętrzne Politechniki Poznańskiej, jak również klientów zewnętrznych.

1.4 Przegląd istniejących rozwiązań

Na rynku dostępny jest szereg gotowych rozwiązań problemu postawionego w tej pracy inżynierskiej. Podobnym projektem jest produkt firmy OPTeam S.A. Jest to niewątpliwie rozwiązanie konkurencyjne, które stosowane jest na innych uczelniach. Umożliwia ono przygotowanie oraz wydawanie kart/legitymacji zgodnych z rozporządzeniami MNiSW. System składa się kilku modułów, które muszą być zainstalowane w całości do poprawnego działania systemu. Moduły te umożliwiają następujące funkcje:

- wydawanie nowych/duplikatów kart,

- personalizację kart,
- zarządzanie cyklem życia kart,
- wprowadzanie danych do kart w sposób ręczny bądź za pomocą zewnętrznych systemów,
- integrację z systemami dostępnymi na uczelni,

oraz wiele więcej. Jest to system bardzo podobny do zaprojektowanego w ramach tej pracy dyplomowej, jednak jako system zewnętrzny wymagałby zmian w dotychczasowej infrastrukturze Działu Rozwoju Oprogramowania oraz ewentualne dostosowywania oprogramowania wymagałyby interwencji producenta.

1.5 Zakres pracy

Celem realizacji projektu jest zaprojektowanie uniwersalnego systemu, który będzie w stanie zarządzać kartami przez cały cykl ich życia. Klient stawia dla produktu jako główny cel zarządzanie elektronicznymi kartami pracowników. Cel pracy dyplomowej został ograniczony do zarządzania legitymacjami pracowniczymi Politechniki Poznańskiej. Oprogramowanie ma zastąpić istniejące rozwiązania, upraszczając i uogólniając je w taki sposób, aby system można było w łatwy sposób przystosować do zarządzania innymi profilami kart oraz rozszerzyć o dodatkowe moduły (np. zarządzanie własną kartą przez użytkowników).

1.6 Zespół realizujący projekt

Promotor:

prof. dr hab. inż. Joanna Józefowska - Zakład Badań Operacyjnych i Sztucznej Inteligencji, Wydział Informatyki, Politechnika Poznańska

Pracownicy Działu Rozwoju Oprogramowania Politechniki Poznańskiej:

mgr inż. Marek Gosławski - pomysłodawca projektu, wsparcie merytoryczne

mgr inż. Tomasz Sawicki - wsparcie techniczne, organizacja środowiska pracy

mgr inż. Mateusz Leszner - wsparcie merytoryczne, testowanie drukowania kart

Menadżerowie:

inż. Dawid Gierszewski - Informatyka - Technologie Wytwarzania Oprogramowania, Wydział Informatyki, Politechnika Poznańska

inż. Tomasz Gramza - Informatyka - Technologie Wytwarzania Oprogramowania, Wydział Informatyki, Politechnika Poznańska (kierownik projektu)

inż. Łukasz Szwajkowski - Informatyka - Technologie Wytwarzania Oprogramowania, Wydział Informatyki, Politechnika Poznańska

Programiści:

Maciej Sobkowiak - Informatyka, Wydział Informatyki, Politechnika Poznańska (lider zespołu programistów)

Paweł Brzeziński - Informatyka, Wydział Informatyki, Politechnika Poznańska

Kamil Czyżnielewski - Informatyka, Wydział Informatyki, Politechnika Poznańska

Dominik Minc - Informatyka, Wydział Informatyki, Politechnika Poznańska

1.7 Podział zadań w zespole programistów

Wielkość i złożoność projektu, wymagała podziału czynności między poszczególnych członków zespołu. W większości przypadków, mogli oni tworzyć kolejne części systemu, niezależnie od siebie.

Ze względu na odbyte praktyki, obejmujące 160 godzin pracy w Dziale Rozwoju Oprogramowania w Politechnice Poznańskiej, liderem zespołu jednogłośnie został wybrany Maciej Sobkowiak. W czasie praktyk miał on okazję zapoznać się bliżej z problemem, co ułatwiło wszystkim rozpoczęcie prac. Rozkład prac nad projektem wyglądał następująco:

Maciej Sobkowiak - lider zespołu

- zaprojektowanie bazy danych,
- wybór aplikacji komunikacyjnej,
- konfiguracja serwera do pracy,
- tworzenie webserwisu aplikacji,
- moduł bazowy SCMS oraz aplikacji klienckiej,
- moduł administracyjny SCMS:
  - organizacje,
  - role,
  - autoryzacja klientów,
- moduł zarządzania kartami:
  - modele blankietow,
  - metody dywersyfikacji,
  - designy,
  - podgląd karty,
  - klucze,
- synchronizacja danych z systemu eKadry.

Paweł Brzeziński:

- moduł zarządzania kartami:
  - aplikacje na karty,
  - zależności między aplikacjami,
  - operacje,
  - pluginy,

  - profile kart,
  - generowanie podglądu karty na podstawie przygotowanych danych,
- logowanie do aplikacji klienckiej poprzez eKonto,
- wtyczka źródła danych dla międzyuczelnianego centrum personalizacji (MCP),
- konsultacje odnośnie testowania wydruku.

Kamil Czyżnielewski:

- tworzenie kart,
- tworzenie zleceń i tak dalej ogólnie ich obsługa,
- modyfikacje layoutów,
- optymalizacja akcji pobierających dużo danych - serwer side do tabelki,
- obsługa,
- pluginy przeglądanie kart,
- przygotowanie aplikacji pod wielojęzyczność

Dominik Minc:

- implementacja mechanizmu słowników
- przygotowanie aplikacji do testów
- operacje na kartach w aplikacji klienckiej
  - zmiany statusów kart
  - usuwanie zleceń
  - obsługa operacji grupowych
- dokumentacja protokołu komunikacyjnego

Rozdział 2

Modelowanie biznesowe systemu

2.1 Aktorzy

Jedynymi zdefiniowanymi aktorami w systemach są:

- Administrator - osoba odpowiedzialna za zarządzanie centralną aplikacją,
- Użytkownik - osoba korzystająca z aplikacji klienckiej,
- Operator - osoba odpowiedzialna za obsługę drukarek kart.

2.2 Obiekty

Obiekty występujące w projektowanym systemie są następujące:

Wzór, wzór graficzny - informacja jaka zostanie naniesiona na kartę w postaci wydruku grafiki. Karta poddawana personalizacji może być czysta (biała) i wtedy końcowy wygląd zależy tylko od wzoru graficznego, ale może też być dostarczona z fabrycznym nadrukiem i końcowy wygląd będzie złożeniem nadruku i wzoru graficznego.

Profil karty - zbiór aplikacji, które będą determinować charakter karty. Każda z nich może definiować zestaw potrzebnych informacji do stworzenia zlecenia personalizacji, dane udostępniane inne aplikacjom, dane tworzące wzór graficzny oraz informacje zapisywane na karcie (chipie, pasku magnetycznym - w zależności od typu karty).

Zlecenie personalizacji - polecenie przeprowadzenia procesu personalizacji karty, który obejmuje naniesienie wzoru graficznego oraz zapis informacji na karcie (zapis danych na chipie, na pasku magnetycznym - w zależności od typu karty). Zlecenie tworzone jest na podstawie profilu, który definiuje ogólne założenia co do wyglądu i funkcjonalności karty. Aby zlecenie było kompletne należy dostarczyć informacje, które wypełnią profil i nadadzą karcie indywidualny charakter.

Karta elektroniczna, karta chipowa (ang. smart card) — uniwersalny nośnik danych w postaci karty wykonanej z plastiku z umieszczonym na niej (lub wewnątrz niej) jednym lub kilkoma układami scalonymi (chip), które pozwalają na ochronę procesu logowania użytkownika, kontrolę dostępu i zawartych na niej danych.

Typ karty, typ fizyczny karty, blankiet - fizyczna niespersonalizowana karta rozumiana jako pusty nośnik, który będzie personalizowany. Każdy typ posiada pewne charakterystyczne dla siebie parametry, które należy uwzględnić podczas tworzenia profilu (m.in. wybierając kompatybilne aplikacje) oraz w procesie drukowania (m.in. rozmiar karty).

2.3 Przypadki użycia

Przypadki użycia można znaleźć w załączniku jakim jest *Dodatek A*.

2.4 Problemy i koncepcja ich rozwiązania

Po wstępnej analizie zadania można wywnioskować, iż komunikacja między systemem centralnym a częścią kliencką będzie kluczowa. Błędnie zaprojektowana może spowodować zagubienie zamówień dla nowych kart, bądź aktualizacji już istniejących. Rozwiązaniem tego problemu jest stworzenia infrastruktury rozproszonej bazującej na przekazywaniu komunikatów między systemami. Idealnym w tym rozwiązaniu okazał się silnik projektu RabbitMQ bazujacego na otwartym protokole AMQP. Wartym podkreślenia jest fakt, iż RabbitMQ zapewnia niezawodność w dostarczaniu wiadomości. RabbitMQ działa w architekturze klient-serwer. Część serwerowa została napisana w języku *Erlang* i dlatego wymaga najnowszej wersji środowiska uruchomieniowego Erlang. Stworzenie aplikacji klienckiej jest ułatwione poprzez liczne przykłady w różnych językach programowania. Stwarza to możliwość tworzenia aplikacji klienckich w różnych technologiach, nie tylko w PHP. Ważnym warunkiem poprawnego działania całej architektury jest odpowiednia wersja RabbitMQ. Wymagane jest aby każda maszyna w rozproszonym systemie posiadała tę samą wersję RabbitMQ.

Następnymi problemami były wymagania postawione przez klienta. Jednym z wymagań było, aby aplikacje: SCMS oraz kliencka odpowiadały na działania użytkownika w czasie nie dłuższym niż 3 sekundy. Przy ogromnych ilościach danych jakie muszą zostać obsłużone w systemach stanowi to dość poważny problem. Dobrym rozwiązaniem okazała się technologia AJAX[1]. Z tej technologii korzysta komponent, który wyświetla dane - *DataTables*. Po przeprowadzeniu prostej konfiguracji komponentu oraz stworzeniu odpowiednich skryptów zostały uzyskane wszystkie zamierzone cele. Wspomniane skrypty wspomagają pracę komponentu, gdyż w danej chwili do komponentu nie są przekazywane wszystkie dane. *DataTables* korzysta z danych, które są przygotowywane po stronie serwerowej i przesyłane do komponentu za pomocą technologii *AJAX*. Dzięki tym zabiegom czas oczekiwania został zniwelowany do minimalnego, który nie przeszkadza w użytkowaniu aplikacji.

Wymagana była również integracja z zewnętrznymi systemami, jak np. eLogin, eKadry. Do rozwiązania tego problemu zostały wykorzystane usługi internetowe, opisane w języku do definiowania usług sieciowych WSDL[2], dostarczone przez DRO. Powyższe

[1] ang. Asynchronous JavaScript and XML - technika tworzenia aplikacji internetowych, w której w sposób asynchroniczny odbywa się interakcja użytkownika z serwerem.

[2] ang. Web Services Description Language

usługi sieciowe korzystają z protokołu SOAP[3]. Jest to protokół wywoływania zdalnego dostępu do obiektów, w tym przypadku obiektów udostępniających dane logowania oraz dane pracowników Politechniki Poznańskiej. SOAP wykorzystuje HTTPS do przenoszenia wywołań. Wszystkie te technologie stworzyły usługi, które w sposób bezpieczny pobierają wymagane dane. Jednakże wielkość danych, np. z eKadr jest bardzo duża, dlatego w celu zapewnienia krótkiego czasu odpowiedzi aplikacji, synchronizacja danych z tej usługi jest uruchamiana w tle.

2.5 Polityka prywatności w ujęciu biznesowym

Celem projektu było również zapewnienie odpowiedniego poziomu bezpieczeństwa, gdyż system będzie operował na danych osobowych pracowników Politechniki Poznańskiej. Podstawą prawną do stworzenia tego dokumentu były:

- Ustawa z dnia 29 sierpnia 1997 r. o ochronie danych osobowych (Dz. U. 1997 nr 133 poz. 883 z późniejszymi zmianami),
- Rozporządzenie Ministra Spraw Wewnętrznych i Administracji z dnia 29 kwietnia 2004 r. w sprawie dokumentacji przetwarzania danych osobowych oraz warunków technicznych i organizacyjnych, jakim powinny odpowiadać urządzenia i systemy informatyczne służące do przetwarzania danych osobowych (Dz. U. 2004 nr 100 poz. 1024)

Wszystkie dane znajdują się w bazie danych *PostgreSQL*, która znajduje się w sieci lokalnej modułu klienckiego SCMS bądź modułu SCMS. Całość znajduje się za zaporami ogniowymi, która gwarantują brak możliwości dostępu do nich osobom nieupoważnionym.

Komunikacja między modułami systemu odbywa się za pomocą aplikacji komunikacyjnej, która gwarantuje niezawodność dostarczania wiadomości. Wszystkie wiadomości wymieniane między systemami są szyfrowane symetrycznym szyfrem blokowym AES[4], który wymaga znajomości odpowiedniego klucza do szyfracji i deszyfracji wiadomości. Klucz ten jest przechowywany w bazie danych w sposób niejawny (jest zaszyfrowany funkcją haszującą). Klucze są prywatne i znane tylko i wyłącznie upoważnionym osobom.

Komunikacja między aplikacjami internetowymi a przeglądarką użytkownika odbywa się za pomocą szyfrowanej wersji protokołu HTTP - HTTPS, co gwarantuje bezpieczeństwo w przekazywaniu kluczowych danych, wprowadzanych przez użytkownika.

[3] ang. Simple Object Access Protocol
[4] ang. Advanced Encryption Standard

Rozdział 3

Specyfikacja wymagań

3.1 Specyfikacja wymagań funkcjonalnych

Przed rozpoczęciem pracy nad projektem przeprowadzona została analiza wymagań funkcjonalnych. Na jej podstawie możliwe było opisanie pożądanego zachowania systemu, stanowiącego klucz do dalszej pracy. Wymagania funkcjonalne określają jakie usługi ma oferować tworzony system, jak ma reagować na określone dane wejściowe, czy też jak ma zachowywać się w określonych sytuacjach. Dodatkowo wśród tych wymagań mogą znaleźć się informację na temat tego, czego system nie powinien robić. Wymagania funkcjonalne zostały przygotowane w postaci tzw. Opowieści Użytkownika (*ang. User Stories*) jak również w postaci przypadków użycia, gdzie zostały pogrupowane ze względu na moduły, których dotyczą.

3.1.1 Specyfikacja dotycząca SCMS

Centralna część systemu SCMS ma za zadanie ułatwić administrowanie całym systemem. Dostęp do niej uzyskuje jedynie administrator z miejsca pracy, czyli sieci lokalnej, co zwiększa poziom bezpieczeństwa. Administrator po pomyślnym uwierzytelnieniu nazwą użytkownika i hasłem, otrzymuje dostęp do aplikacji, którą ze względu na dużą złożoność można podzielić na moduły.

Jedną z części SCMS jest tzw. **moduł administracyjny**. Odpowiada on za zarządzanie organizacjami, ich rolami w systemie jak również autoryzacją aplikacji klienckich. Głównym celem tego modułu jest możliwość dodawania w łatwy sposób kolejnych klientów do systemu. Administrator może określić nazwę klienta i nadać odpowiednie uprawnienia.

Druga część SCMS — **moduł zarządzania kartami** — umożliwia administratorowi w łatwy sposób sprawować kontrolę nad dostępnymi w systemie kartami.

Administrator ma możliwość:

- tworzenia nowych profili kart dla klienta. Profil to rodzaj karty elektronicznej, rozumiany jako np. legitymacja studencka, czy też legitymacja pracownicza. Podczas definiowania nowego profilu Administrator ma możliwość powiązania go z organizacjami, które będą mogły z niego korzystać, jak również z modelami blankietów,

z którymi profil będzie kompatybilny,

- dodawania do systemu nowych aplikacji na karty. Aplikacja to oprogramowanie definiujące funkcjonalność danej karty np. aplikacja umożliwiająca dostęp do parkingu przy uczelni,
- wyszukiwania spośród dostępnych aplikacji oraz dodawania ich do profilu za pomocą pojedynczego kliknięcia,
- wyświetlania aktualnie znajdujących się w profilu aplikacje na karty oraz zależności pomiędzy nimi,
- modyfikowania zależności między aplikacjami dla danego profilu,
- dodawania nowego wyglądu karty, który został dostarczony przez klienta w postaci obrazu. Obraz ten może zostać dodany do profilu karty klienta,
- edytowania wyglądu karty, oraz generować jej podgląd przed wydrukiem.

Istotną częścią systemu SCMS jest **Aplikacja do drukowania kart**. Dane niezbędne do wydrukowania karty pobierane są z bazy danych systemu SCMS. Dzięki specjalnie zaprojektowanej wtyczce programowej drukarka otrzymuje dane jakie ma nanieść na blankiety.

3.1.2 Specyfikacja dotycząca aplikacji klienckiej

Drugim elementem systemu jest aplikacja kliencka. Dostęp do niej uzyskuje operator zarówno przez istniejące konto eLogin, jak również poprzez logowanie z wykorzystaniem odrębnego konta, dedykowanego dla tej aplikacji klienckiej

Jedną z części Aplikacji klienckiej jest tzw. **moduł zleceń**.
Operator aplikacji klienckiej wystawia nowe karty (dla pracowników przyjętych od nowego roku akademickiego). Operator aplikacji klienckiej może w prosty sposób zlecić wystawienie kart pracowniczych wszystkim pracownikom, którzy nie mają jeszcze karty (np. zostali dopiero przyjęci). Ma on dostęp do bazy pracowników, która synchronizuje się z zewnętrznym system. Operator ma możliwość dodawania zdjęć pracowników, wybrania profilu karty dla nich oraz podawania innych potrzebnych danych do wysłania zlecenia personalizacji. Kluczowe w tym wypadku jest pokazanie pracowników, którzy nie mają jeszcze karty.

Operator aplikacji klienckiej może sprawdzić, kiedy dostanie zlecone karty. Weryfikuje to na podstawie statusu zamówionych kart. Informacje te są niezbędne do sprawnego zarządzania kartami i dostarczenia ich do użytkowników końcowych w określonym czasie.

Operator aplikacji klienckiej, ma możliwość blokowania karty klienta w przypadku jej zagubienia przez właściciela.

Operator może, w razie potrzeby, obejrzeć historię kart pracownika. Od kiedy do kiedy były ważne dane karty, jakie były ich numery, wyglądy i profile. W razie potrzeby może również wydać duplikat danej karty oraz ją unieważnić.

Inną częścią Aplikacji klienckiej jest tzw. **moduł self-service**.

Użytkownik może szybko uzyskać dostęp do systemu zarządzającego swoją kartą. System do zarządzania kartą jest połączony z istniejącym obecnie systemem w firmie/uczelni użytkownika, tak aby użytkownik nie musiał pamiętać kolejnego adresu oraz danych do logowania.

3.1.3 Specyfikacja dotycząca wtyczki do drukowania

Następnym elementem pracy dyplomowej było stworzenie wtyczki pobierającej dane z bazy danych SCMS w celu przesłania ich do modułu drukującego. Wtyczka ta została napisana w języku C#.

3.2 Specyfikacja wymagań pozafunkcjonalnych

Poza analizą wymagań funkcjonalnych na rozwiązanie wpływają również wymagania pozafunkcjonalne, nie związane bezpośrednio z funkcjami aplikacji. Opisują one przede wszystkim własności produktu dotyczące interfejsu użytkownika, jego cechy i charakterystyki. Precyzują jakie interfejsy sprzętowe i programowe będą użyte do tworzenia systemu. Regulują również kwestie wydajności (czas odpowiedzi systemu), czy też niezawodności (czas bezawaryjnej pracy), jak również bezpieczeństwa tworzonego systemu oraz poufności przetwarzanych danych. Opracowanie wymagania pozafunkcjonalne stanowią swego rodzaju listę zasad, od których nie można odbiegać podczas realizowania projektu.

3.2.1 Kryteria i standardy jakości

Interfejsy użytkownika
Wymagane jest, aby zarówno SCMS jak i aplikacja kliencka posiadały interfejs webowy (cienki klient) - w formie stron www. Graficzny interefejs użytkownika musi być spójny z dotychczasowymi rozwiązaniami oprogramowania Politechniki Poznańskiej (np. eLogin, eStudent 2.0). Szczegółowe wytyczne opisuje standard: SIW/CI PP. Strony www będą przygotowane w sposób wspierający wielojęzyczność. Przygotowane będą treści w języku polskim. Ponadto przygotowana zostanie aplikacja drukowania w postaci aplikacji lokalnej (gruby klient). Dla formularzy w których pojawia się lista elementów, na których wykonywane są operacje, które mogą być wykonywane grupowo - powinna istnieć możliwość grupowego wykonania tych operacji. Nawet jeżeli dopuszczono odstępstwo w tym zakresie to sposób wywołania funkcji wykonującej dana operację powinien umożliwiać jej wywołanie dla wielu wierszy listy. Dla widoku list powinna istnieć wprost możliwość sortowania list po kolumnach wyświetlanych na liście. Dla widoku list powinna byc możliwość wyszukiwania po zaawansowanych kryteriach - kryteria powiiny być możliwe do ustalenia przez administratora systemu lub użytkownika. Dla widoku list powinna istnieć możliwość eksportu zawartości do plików edytowalnych (np. xls, csv)

Interfejsy sprzętowe
Wymagane jest, aby moduł drukowania współpracował z urządzeniami specjalistycznymi do obsługi kart. Podczas wykonywania personalizacji karty wykonywany będzie na karcie nadruk graficzny oraz nastąpi wprowadzenie danych do pamięci układu elektronicznego.

Wszystkie niezbędne dane będą przekazywane z modułu drukowania do drukarki poprzez odpowiednią wtyczkę programu. Wtyczka ta będzie wykorzystywać sterownik dostępnej drukarki.

Interfejsy programowe
Aplikacja kliencka odpowiada m.in. za pozyskanie danych z zewnętrznego systemu eKadry. [9]. Dane pobrane z zewnętrznego systemu będą przeniesione do bazy danych połączonej z modułem personalizacji. W celu uwierzytelnienia użytkownika wykorzystanie zostany zewnętrzny system eKonto [10].

Wydajność
Należy zapewnić odpowiedni poziom szybkości działania aplikacji. Przy założeniu 30 równocześnie aktywnych użytkowników na standardowej maszynie serwerowej (Xeon E3500, 8GB RAM, Linux, dwa łącza 100Mbps), czas odpowiedzi powinien wynieść maksymalnie 10 sekund w 90% przypadków. Jeżeli czas odpowiedzi musi zostać przekroczony, należy powiadomić użytkownika o spodziewanym wydłużonym czasie obsługi żądania. Nie jest dopuszczalna realizacja żądania w ramach sesji http w czasie dłuższym niż 1 minuta.

Bezpieczeństwo
Wymagane jest, aby system spełniał wymagania ustawy o ochronie danych osobowych. Krytyczne operacje powinny być dodatkowo potwierdzane przez użytkownika. Wymagane jest, aby oprócz normalnego trybu pracy systemu, istniała możliwość działania systemu w następujących trybach.

- Tryb testowy, w którym działania w systemie nie powodują oddziaływania na otoczenie systemu (system operuje na testowych danych).
- Tryb maintenance, w którym następuje blokada dostępu dla użytkowników oraz wyświetlana jest informacja, że w systemie trwają prace konserwacyjne.

Nie jest wymagane, by system był odporny na awarie. System musi natomiast w przypadku wystąpienia awarii poinformować o niej użytkownika w sposób kontrolowany. System musi zapewnić odpowiedni poziom tolerancji uszkodzeń i odtwarzalności zbioru danych aplikacji. Należy zapewnić trwałość danych w przypadku awarii łączy z dokładnością do transakcji, a w przypadku awarii serwerowych nośników danych - z dokładnością do godziny (zmiany wprowadzone wcześniej niż godzinę przed momentem wystąpienia awarii nie mogą zostać utracone). Żadna awaria nie może też powodować naruszenia wymagań bezpieczeństwa i poufności.

Poufność
Komunikacja z systemem będzie realizowana za pomocą szyfrowanej wersji protokołu HTTP - HTTPS.

Jakość
Wymagana jest kompatybilność z następującymi przeglądarkami internetowymi: Google Chrome, Firefox, Internet Explorer, Opera.
W odniesieniu do jakości dokumentacji projektu, wymagane jest stosowanie się do określo-

nych szablonów dokumentów. W odniesieniu do jakości kodu źródłowego, wymagane jest stosowanie się do określonych standardów kodowania i nazewnictwa obiektów bazy danych.

Inne wymagania
Przenośność / łatwość instalacji — W celu zapewnienia przenośności aplikacji, dopuszczalne jest zastosowanie jedynie komponentów z otwartym kodem źródłowym, których licencja nie przewiduje odpłatności przy jakichkolwiek zastosowaniach, w tym także komercyjnych.
Należy dostarczyć dokument instalacji systemu, który pozwoli osobie nie zaangażowanej w projekt, dysponującej już skonfigurowanym środowiskiem (zainstalowane odpowiednie oprogramowanie i skonfigurowany system operacyjny) wdrożenie projektu w przeciągu 3 dni.

3.2.2 Standardy kodowania

Standard kodowania może obejmować wiele aspektów kodu programu: Standardy kodowania to zasady wprowadzane w celu ujednolicenia wyglądu i działania tworzonego kodu.

Dotyczą one wielu aspektów takich jak np.:

Formatowanie kodu

- szerokość wcięcia,
- maksymalna długość wiersza,
- liczba pustych wierszy między kolejnymi definicjami i deklaracjami funkcji, klas.

Konwencje nazewnicze

- schemat nazywania funkcji, klas, zmiennych, modułów, przestrzeni nazw, plików itp.

Komentowanie kodu

- sposób komentowania kodu - rodzaj komentarzy,
- ilość komentarzy oraz ich złożoność,
- konieczność udokumentowania użytych algorytmów,
- styl komentowania - np. zgodny z narzędziem automatycznie generującym dokumentację.

Konstrukcje programistyczne

- zabraniane polecane i konstrukcje,
- zalecane konstrukcje dla konkretnych problemów.

Podobnie jak zachowanie aplikacji, część logiczna także powinna się stosować do określonych zasad. Klasy, funkcje i zmienne powinny być nazwane w logiczny sposób tak, aby ich nazwy dobrze oddawały intencję programistów. Tworzony kod powinien nie tylko zapewniać odpowiednie działanie systemu, lecz również sam siebie opisywać. Oto zestawienie

zysków i strat wynikających ze stosowania się przez programistów do standardów kodowania:

Zalety:

- łatwość zlokalizowania odpowiedniego miejsca w kodzie w przypadku awarii,
- łatwość zrozumienia kawałku kodu wyjętego z kontekstu,
- łatwość wdrożenia nowych programistów do pracy przy projekcie,
- osoby uczące się dopiero danego języka programowania, nabierają dobrych praktyk podczas pracy z kodem,
- mniejsza ilość błędów w kodzie,
- przejrzysty kod.

Wady:

- redukują kreatywność,
- wydłużenie czasu pracy,
- są zbędne tak długo jak programiści tworzą kod w skupieniu, z dobrym przemyśleniem,
- często wymagają wiele dodatkowych struktur w kodzie,
- niektórzy programiści ignorują standardy, co wprowadza rozbieżności.

Standardy kodowania są ważne w każdym projekcie programistycznym, a już szczególnie, gdy przy tym samym projekcie pracuje większa liczba programistów. Standardy kodowania pomagają zapewnić wysoką jakość kodu, mniejszą liczbę błędów i łatwe zarządzanie.

Przed przystąpieniem do tworzenia aplikacji, każdy programista z reguły planuje stosować pewne reguły dotyczące tworzonego kodu. Jednak kiedy projekt się rozpoczyna, najczęściej dobre założenia zostają porzucane jedno po drugim. Po zakończonych pracach, bardzo często okazuje się, że kod jest mało zrozumiały nawet dla jego twórców.

Ze względu na fakt, iż tworzone programowanie będzie dalej rozwijane postawiono programistom wymóg zaakceptowania i stosowania się do poniższej specyfikacji [7].

Wcięcia

1. Wcięcia wykonane za pomocą tabulatora.
2. Tabulacje prezentowane w postaci 4 spacji.

Bloki

1. Nawias otwierający umieszczamy w tej samej linii co początek bloku.
2. Nawias zamykający umieszczamy jako jedyny znak w linii na tej samej wysokości co początek bloku.

Spacje

1. Jedna spacja po instrukcji kontrolnej (if, for, foreach, while, itd.)
2. Bez spacji po nawiasie otwierającym i przed nawiasem zamykającym (dotyczy nawiasów () i []). początek bloku (z kilkoma wyjątkami).
3. Jedna spacja przed i po operatorze (|| . ==, +, -, itd.)
4. Jedna spacja po przecinku.

Struktury kontrolne

if

```
if ($var == 2)
    print '2';
```

```
if ($var == 2) {
    print '2';
}
```

```
if ($var == 2) {
    print '2';
}
else {
    print 'roĹźne od 2';
}
```

```
if ($var == 2) {
    print '2';
}
elseif ($var == 3) {
    print '3';
}
else {
    print 'ani 2, ani 3';
}
```

Dopuszczalne jest też doklejanie else do nawiasu zamykającego:

```
if ($var == 1) {
    print '1';
} elseif ($var == 2) {
    print '2';
} else {
    print 'ani 1, ani 2';
}
```

while ,do

```
while (true) {
    trigger();
}
```

```
do {
    trigger();
} while (true);
```

switch

```
switch ($var) {
    case 1:
        print '1';
        break;
    case 2:
        print '2';
        break;
    default:
        print 'ani 1, ani 2';
        break;
}
```

for

```
for ($j = 1; $j < 5; $j++) {
    print $j;
}
```

foreach

```
foreach ($array as $key => $value) {
    print $value;
}
```

try..catch

```
try {
    //kod testowany
}
catch (Exception $e) {
    //obsluga wyjatku
}
```

Zmienne

1. Słowa oddzielone znakiem podkreślenia.
2. Małe litery.

```
$my_favourite_variable = 1;
```

Procedury

1. Nazwa w formacie slowoSlowoSlowo.

```
function myFunction($param1, $param2) {
    print $param1 . $param2;
}
```

Klasy

1. Nazwa w formacie SlowoSlowoSlowo.

```
class MyClass {
    private $my_priv;

    public $my_pub;

    function __construct() {
        //kod konstruktora
    }

    public function myMethod($param1, $param2) {
        print $param1 . $param2;
    }
}
```

Tablice

1. Większe tablice można rozbijać w celu większej czytelności.

```
$my_array = array (
    'foo'  => 'bar',
    'spam' => array (
        'one' => 1,
        'two' => 2
    )
);
```

Inne

1. Nie jest wymagane stosowanie zawijania wierszy. Jednak w przypadku jego stosowania należy zawijać po 160tym znaku.
2. Nazwy zmiennych, klas, metod, funkcji, stałych itd. powinny być w języku angielskim.
3. Preferowanie użycia apostrofu zamiast cudzysłowiu do konstrukcji ciągów tekstowych.
4. Preferowanie użycia sprintf zamiast konkatenacji kropką przy złożonych połączeniach ciągów tekstowych.
5. Zalecenie używania tagu <? zamiast <?php

6. Nie umieszczanie zamykającego tagu ?> na końcu pliku PHP. Koniec pliku to dla PHP to samo co koniec skryptu, a nie ma problemów z wiszącymi znakami po zamykającym tagu.
7. Przy mieszaniu skryptu z HTML ograniczyć używanie instrukcji print/echo.

```
<?
if ($value == 1) {
?>
    <a href="#">1</a>
<?
} else {
?>
    <a href="#">2</a>
<?
}
?>
```

lub

```
<? if ($value == 1) { ?>
    <a href="#">1</a>
<? } else { ?>
    <a href="#">2</a>
<? } ?>
```

zamiast

```
<?
if ($value == 1) {
    print '<a href="#">1</a>';
else
    print '<a href="#">2</a>';
?>
```

8. Do wypisywania pojedynczych wartości w środku kodu HTML używać konstrukcji

```
<?=$value?>
```

zamiast

```
<? echo $value; ?>
```

9. Nie bać się wyrażenia

```
<?=warunek ? 'tekst1' : 'tekst2'?>
```

3.2.3 Wymagania dotyczące dokumentacji

Istotną część dokumentacji będą stanowić komentarze do kodu tworzące na bieżąco w trakcie pracy nad aplikacją. Komentarze kodu powinny być formatowane zgodnie z phpDocumentor [1]. Zalecane jest stosowanie wszystkich tagów zgodnie ze specyfikacja i uznaniem programisty. Niemniej jednak wymagane są:

- dla klasy:
  - krótki opis,
  - @author - autor (autorzy).
- dla atrybutu klasy:
  - krótki opis,
  - @var.
- dla metody klasy:
  - krótki opis,
  - @param dla każdego parametru,
  - @return,
  - @throws.a

Zalecane jest stosowanie (zdroworozsądkowe) tagu *@see* [2], aby ułatwić poruszanie się po dokumentacji.

Rozdział 4

Architektura systemu

Zasadniczo budowa minimalnego wariantu systemu jest prostą architekturą rozproszoną. System składa się z trzech podstawowych komponentów:

- Centralnej aplikacji SCMS, zawierającej informacje oraz dodatkowe oprogramowanie niezbędne do przetworzenia zamówienia i stworzenia fizycznej karty, wraz ze stanowiskami drukującymi.
- Aplikacji klienckiej, pozwalającej na wysyłanie zamówień do aplikacji centralnej, śledzenie statusów poszczególnych kart oraz dokonywania na niej zmian
- Aplikacji komunikacyjnej, która pośredniczy w wymianie komunikatów między elementami architektury (SCMS i aplikacja kliencka posiadają własne moduły komunikacyjne).

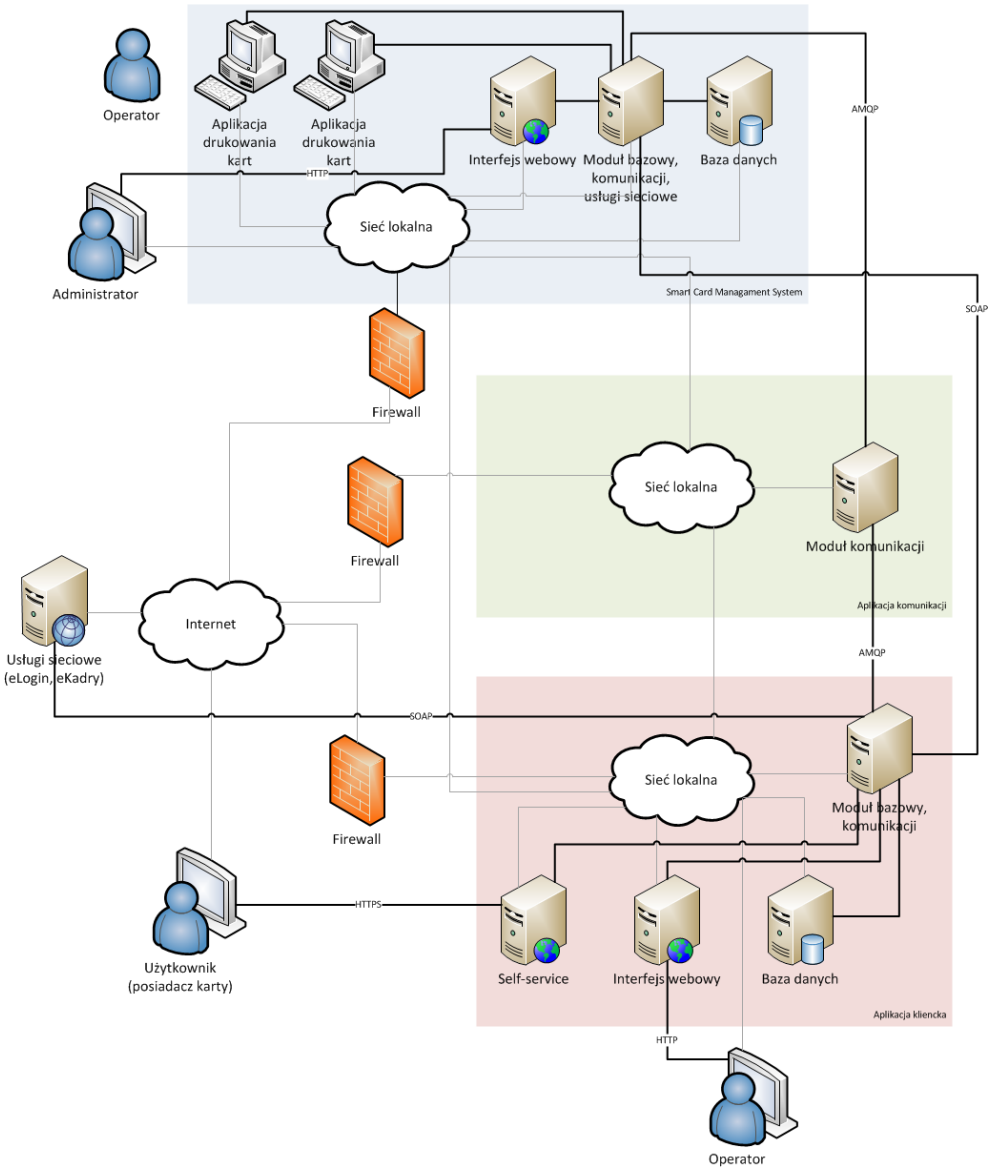
Na potrzeby realizacji kart elektronicznych spersonalizowanych w Politechnice Poznańskiej w powyższej architekturze należało dodatkowo uwzględnić zewnętrzne źródła danych, które pobierane będą poprzez aplikacje klienckie, co pozwoli na ich dalsze przetwarzanie przez system. Źródłami danych w obecnym etapie prac są:

- System eLogin - logowanie pracowników i studentów do aplikacji klienckiej za pomocą zintegrowanego systemu uwierzytelniania stosowanego na uczelni. Znajdą tam oni informacje dotyczące ich kart oraz dostępne operacje w ramach modułu self-service
- System eKadry – synchronizacja danych pracowników do aplikacji klienckiej. Dane te pozwalają na dokonywanie zleceń personalizacji (tworzenie lub duplikat) kart.

Uzupełnieniem architektury są zapory ogniowe, ograniczające dostęp do systemu tylko z zaufanych podsieci oraz blokujące przed znanymi typami ataków na serwery sieciowe.

Całość architektury prezentuje Rysunek 1.:

Obecna konstrukcja systemu dopuszcza dodawanie dodatkowych źródeł danych, których wiadomości zgodne są z architekturą SOAP (usługa sieciowa implementowana przez wiele języków programowania) oraz protokołem AMQP (zgodność z aplikacją komunikacyjną, dostępne klienty w wielu językach). Należy jednak zaimplementować obsługę tych dodatkowych źródeł danych.



Rysunek 4.1: Architektura systemu

4.1 SCMS

W ramach systemu SCMS zaimplementowane zostały następujące moduły:

- moduł bazowy – obsługa logiki bazy danych, możliwość logowania,
- moduł administracyjny (Interfejs webowy) – możliwość definiowania profili kart, modeli blankietów, designów oraz autoryzacji aplikacji klienckich,
- moduł komunikacji – odbieranie komunikatów adresowanych do SCMS wysyłanych przez aplikacje klienckie z aplikacji komunikacyjnej jak i również wysyłanie komunikatów (możliwość transmisji broadcastowej, multicastowej oraz unicastowej),
- usługi sieciowe – udostępnianie metod poprzez SOAP dla aplikacji klienckich. Służy do operacji wymagających mniejszej ilości danych (umożliwia sprawdzenie autoryzacji danej aplikacji klienckiej, pobranie listy profili oraz jego instalację na aplikacji klienckiej). Metody obsługiwane przez usługi sieciowe można także zaimplementować w module komunikacyjnym rozszerzając go o nowe typy komunikatów,

System SCMS posiada swoją bazę danych, którą współdzieli ze stanowiskami drukującymi. Pozwala to na pobranie informacji o karcie lub zamówieniu bezpośrednio z bazy danych do stacji drukującej oraz zmianę statusów karty w czasie rzeczywistym.

4.2 Aplikacja kliencka

Aplikacja kliencka, podobnie jak SCMS, została podzielona na moduły:

- moduł bazowy – logowanie oraz logika bazy danych,
- moduł administracyjny (interfejs webowy) – synchronizacja danych oraz dokonywanie zleceń personalizacji przez administratora aplikacji klienckiej,
- moduł profilu – zawiera implementacje akcji charakterystyczne dla danego profilu karty. Aplikacja kliencka może posiadać więcej niż jeden moduł profilu, jednak należy go zaimplementować. W chwili obecnej dostępna jest większość funkcji modułu profilu karty pracownika PP,
- moduł komunikacji – odbieranie oraz wysyłanie komunikatów do centralnego systemu SCMS przez aplikację komunikacyjna,
- moduł self-service – umożliwia studentowi lub pracownikowi dokonywanie zmian na już istniejącej karcie oraz przeglądanie kart. Moduł ten został zaimplementowany tylko w niewielkiej części, gdyż pełna funkcjonalność wykraczała poza zakres pracy inżynierskiej.

Aplikacja kliencka również posiada własną bazę danych, której nie współdzieli z innymi systemami.

4.3 Komunikacja pomiędzy SCMS a aplikacją kliencką

Komunikacja między elementami architektury odbywa się na dwa sposoby:

- Poprzez aplikację komunikacyjną RabbitMQ – wykorzystywaną do przesyłania zleceń i odbioru statusów kart. Moduły komunikacyjne elementów architektury (na chwilę obecną posiadają je SCMS oraz aplikacja kliencka) wysyłają komunikaty wraz z informacjami uwzględniającymi odbiorcę (lub odbiorców) komunikatu. Wiadomości przesyłane są zgodnie ze specyfikacją protokołu AMQP (wykorzystana biblioteka php-amqplib [5]), które następnie wędrują do aplikacji komunikacyjnej. Aplikacja ta przekierowuje komunikat do odpowiedniej kolejki, którą obsługuje moduł komunikacyjny odbiorcy wiadomości. Odbiór wiadomości następuje z porządkiem FIFO [6]. Komunikaty wysyłane tą drogą nie zostaną zagubione nawet w przypadku awarii aplikacji komunikacyjnej (mechanizm odtwarzania stanu [6]) i mają gwarancję dostarczenia (mechanizm acknowledge [6]). Aplikacja komunikacyjna obsługuje również mechanizm transakcji (operacje begin transaction, commit oraz rollback [6]), które nie są obecnie wykorzystywane w transmisji danych.
- Komunikacja poprzez usługę sieciową SOAP – wykorzystywana przez aplikację kliencką do synchronizacji profili kart i testu autoryzacji klienta (usługa własna, udostępniania przez SCMS) oraz synchronizacji danych z systemów zewnętrznych i logowania przez eKonto (biblioteki klienckie udostępnione przez DRO). Kanał ten nie gwarantuje niezawodności komunikacji. Został jednak zaimplementowany ze względu na implementację źródeł danych w Politechnice Poznańskiej w standardzie SOAP (eKadry oraz eLogin).

4.4 Schemat bazy danych

Schematy bazy danych dla SCMS i aplikacji klienckiej były początkowo wspólne dla obu aplikacji (schemat ten powstał podczas odbywania praktyk w DRO). Obejmowały one tylko podstawowe ich zastosowania. W trakcie implementowania funkcjonalności w semestrze inżynierskim w obu schematach pojawiały się nowe tabele, charakterystyczne dla każdej z części systemu.

Baza danych uwzględnia szczególny nacisk na modułowość profili - system musi obsługiwać praktycznie dowolny typ karty elektronicznej. Aby było to możliwe, należało zdefiniować ponad 40 tabel zawierających dane oraz wiele połączeń między nimi, wszelkiego rodzaju (1 do 1, 1 do n, m do n). Schemat ten bazowany był na wstępnym schemacie bazy danych do projektu karty PEKA, który został udostępniony przez DRO [8].

Schemat bazy danych został załączony w postaci plików dostępnych na płycie CD.

Rozdział 5

Opis implementacji

5.1 Narzędzia

Użyto systemu zarządzania projektami Redmine połączonego z systemem kontroli wersji Subversion 1.6.17. Bazę danych zaprojektowano za pomocą EMS SQL Manager for PostgreSQL. Dalsze prace nad bazą danych wykonywane były poprzez aplikację phpPgAdmin 5.0.4. Podczas programowania korzystano z następujących środowisk programistycznych: Eclipse, Notepad++ oraz Netbeans. Testy funkcjonalne stworzono poprzez framework Selenium i środowisko Eclipse.

5.2 Technologia

Webowa część aplikacji została zaimplementowana w języku PHP 5.4 w oparciu o framework PHPLiteMVC. Serwer korzysta z wydajnego interpretera nginx wraz z wbudowanym silnikiem proxy i modułem https. Systemem zarządzania bazą danych jest PostgreSQL wspierany przez framework. Po stronie użytkownika, działanie aplikacji wspomaga biblioteka jQuery 1.9 Część lokalna stworzona została w języku C# oraz frameworku .NET w wersji 4.0. Moduł komunikacyjny korzysta z oprogramowania RabbitMQ zgodnego ze standardem protokołu AMQP.

5.3 Harmonogram projektu

Kamień milowy I - Autoryzacja i środowisko testowe: 02.11.2012

Kamień milowy II - Zarządzenie zleceniami: 30.11.2012

Kamień milowy III - Drukowanie: 16.12.2012

5.4 Realizacja zadań

Praca nad systemem rozpoczęła się od spotkań pod kierownictwem mgr inż. Marka Gosławskiego - kierownika Działu Rozwoju Oprogramowania oraz prof. dr hab. inż. Joanny Józefowskiej z zespołami menadżerów i programistów. Spotkania miały na celu określenie funkcjonalności aplikacji, ogólnego sposobu ich implementacji oraz rozwiązania najbardziej znaczących problemów.

Ze strony programistów, realizację poszczególnych zadań poprzedzono zapoznaniem się z frameworkiem PHPLiteMVC oraz powiązanymi z nim technologiami. Zespół menadżerów odpowiedzialny był za przygotowanie user stories, przypadków użycia oraz przydział zadań do poszczególnych kamieni milowych. Zarządzanie projektem wspomagane było poprzez środowisko Redmine połączone z systemem kontroli wersji SVN, co znacznie ułatwiło komunikację między zespołem projektowym oraz zabezpieczyło przed niechcianymi zmianami w kodzie.

Realizację zadań usprawniały cotygodniowe spotkania zespołu projektowego, na których omawiano postępy w pracy, napotkane problemy oraz przyszłe zadania. Programiści dodatkowo organizowali własne spotkania, na których omawiali trudności implementacyjne danych zagadnień oraz dokonywali podziału prac.

5.5 Podsumowanie realizacji

Implementacja aplikacji rozpoczęła się w październiku 2012 roku i trwała niecałe cztery miesiące. Pierwszy kamień milowy został w pełni zrealizowany na czas, w drugim doszło do niewielkiego opóźnienia. Najwięcej ponadplanowego czasu poświęcono na trzeci kamień milowy, w którym znajdowały się stosunkowo trudne zagadnienia do realizacji, oraz w którym pojawiły się zadania początkowo nieuwzględnione.

Ostatecznie wszystkie zagadnienia zostały w pełni zrealizowane a system został przekazany do użytku i dalszego rozwoju Działowi Rozwoju Oprogramowania Politechniki Poznańskiej.

5.6 Dokumentacja

Dokument przekazania systemu informatycznego do eksploatacji

Zawiera przeznaczenie i krótki opis systemu z uwzględnieniem jego krytyczności. W dokumencie opisana została perspektywa logiczna modułów, wykorzystana technologia oraz minimalne wymagania sprzętowe.

Instrukcja zarządzania systemem informatycznym

Zawiera opis metod i procedur wymaganych przez ustawę dotyczącą przetwarzania danych osobowych.

Polityka bezpieczeństwa

Zawiera krótki opis systemu, powiązanych z nim struktur fizycznych oraz zawartość poszczególnych pól w bazie danych, które związane są z przechowywaniem danych osobowych. Dokument wymagany przez ustawę o ochronie danych osobowych.

Dokumentacja protokołu komunikacyjnego

Opis zasady działania protokołu komunikacyjnego oraz struktury i typów komunikatów.

Instrukcja obsługi RabbitMQ dla administratora

Instrukcja dotycząca instalacji i konfiguracji aplikacji komunikacyjnej oraz metody wykorzystania jej w języku PHP.

Dokumentacja usługi sieciowej

Opis struktury usługi sieciowej systemu SCMS. Zawiera metody uwierzytelniające systemy klienckie oraz pozwalające pobrać profile kart z systemu centralnego.

Schemat bazy danych SCMS

Schemat relacji pomiędzy poszczególnymi tabelami oraz zawartych w nich pól.

Schemat bazy danych aplikacji klienckiej

Schemat relacji pomiędzy poszczególnymi tabelami oraz zawartych w nich pól.

Dokumenty znajdują się na płycie CD dołączonej do pracy dyplomowej.

Rozdział 6

Testy

Testowanie jest bardzo istotnym elementem procesu wytwarzania oprogramowania, pomaga zapewnić odpowiednią jakość końcowego produktu. Testowanie aplikacji opiera się przede wszystkim na testach jednostkowych, badających prawidłowe działanie poszczególnych fragmentów kodu oraz funkcjonalnych, sprawdzających poprawność kluczowych elementów programu. Ponadto w aplikacjach webowych istotnym elementem jest responsywność systemu oraz niskie zużycie zasobów pamięciowych, z tego względu przeprowadza się testy wydajnościowe.
Oprócz testowania, przez programistów wykonywane są wzajemne inspekcje, które pomagają sprawdzić poprawność fragmentów kodu i przyspieszają rozwiązanie trudnych problemów.
W webowych aplikacjach bazodanowych wskazane jest również, aby testy dokonujące zmian w bazie danych, uruchamiane były na specjalnie przygotowanej do tego instancji.

6.1 Testy jednostkowe

Testy jednostkowe są pierwszymi testami jakie wykonuje się podczas wytwarzania oprogramowania. Weryfikują one poprawność pojedynczych elementów (jednostek) programu, poprzez porównywanie oczekiwanych wartości (efektów działania) z faktycznym wynikiem.

W trakcie implementacji aplikacji, zrezygnowano z przeprowadzenia powyższych testów ze względu na opóźnienie w przygotowaniu środowiska testowego, a następnie niewystarczającej ilości czasu potrzebnej na wytworzenie poszczególnych instancji testowych.

W ramach zadania przystosowano aplikację do uruchomienia w różnych trybach działania, między innymi w trybie testowym, który korzysta z osobnej bazy danych.

6.2 Testy funkcjonalne

Testy funkcjonalne przeprowadzane są aby określić czy stworzone elementy systemu spełniają oczekiwania klienta, z tego względu ich przeprowadzeniem powinny zajmować się osoby niezwiązane z implementacją danej aplikacji a nawet niepotrafiące programować.

Dla systemów dostępnych przez przeglądarkę internetową, testy tego typu są proste w automatyzacji i istnieją przygotowane w tym celu narzędzia. Jednym z dostępnych tego typu rozwiązań jest zintegrowane środowisko Selenium, w którym działania użytkownika

mogą zostać zapisane i na ich podstawie powstaje gotowy kod jUnit możliwy do uruchomienia np. w programie Eclipse.

Testy przygotowano przez programistów do późniejszej ich edycji i uruchomienia przez Dział Rozwoju Oprogramowania. Dla zwiększenia wiarygodności, poszczególne osoby testowały elementy, w których same nie brały udziału w implementacji. Ze względu na powtarzający się proces autoryzacji, przygotowano tryb testowy dokonujący automatycznego logowania i uruchamiany na oddzielnej bazie danych.

Efektem końcowym było całkowite pokrycie przypadków użycia i spełnienie oczekiwanych reakcji systemu, na podstawie przeprowadzonych działań użytkownika.

6.3 Testy wydajnościowe

Testy wydajnościowe przeprowadzane są w celu oceny działania systemu przy dużym obciążeniu. Dla aplikacji internetowych najczęściej bada się resposywność na wykonane akcje, wpływ wielkości obsługiwanych danych oraz efekty wykorzystania systemu przez zwiększoną liczbę użytkowników.

Zgodnie z wymaganiami pozafunkcjonalnymi, czas reakcji systemu powinien wynosić maksymalnie 3 sekundy, przy sprawnym połączeniu internetowym, z tego względu już w trakcie implementacji zwracana była uwaga na szybkość reakcji. Jedynym elementem systemu, który nie spełnił tego wymogu była synchronizacja danych ze źródeł zewnętrznych - czas odpowiedzi wynosił średnio 7 minut, dlatego też synchronizacja została zaprogramowana do działania w tle. Dla pozostałych elementów testy zakończyły się pozytywnie.

Sam serwer poprzez działanie SCMS oraz aplikacji klienckiej nie wskazywał na znaczące zużycie zasobów zarówno w spoczynku jak i podczas wykonywania akcji. Przy biernym działaniu, wykorzystanie procesora było bliskie zeru a pamięci nie przekraczało kilku procent. Dla najbardziej złożonej operacji jaką jest synchronizacja danych z systemu eKadry, zużycie procesora wzrosło o 0,8%, natomiast pamięci o 2,1%.

Nie zbadano wpływu ilości danych na działanie modułu komunikacyjnego RabbitMQ - ze względu na małe rozproszenie aktualnie komunikujących się elementów, testy nie byłyby wiarygodne.

Rozdział 7

Wnioski i zalecenia wdrożeniowe

7.1 Plan wdrożenia oprogramowania

Wdrożenie aplikacji nie było częścią tej pracy dyplomowej. Po zakończeniu procesu implementacji, kody źródłowe wraz z całą wiedzą merytoryczną zostaną przekazane do Działu Rozwoju Oprogramowania. Następnie aplikacje zostaną prawdopodobnie wdrożone przez specjalistów ze wspomnianego już Działu Rozwoju Oprogramowania.

7.2 Uwagi dotyczące użytkowania systemu

Ważną uwagą dotyczącą użytkowania systemu jest odpowiednia wersja przeglądarki internetowej. Przeglądarka ta obowiązkowo powinna obsługiwać technologię *JavaScript*.

Aplikacje zostały tak zaprojektowane, aby działały pod każdą z dostępnych przeglądarek internetowych. Jednakże, dla zachowania poprawnej szaty graficznej, poprawnego działania wszystkich komponentów warto zaktualizować swoją przeglądarkę do najnowszej wersji, na chwilę dzisiejszą są to:

- Internet Explorer 9
- Mozilla Firefox 18
- Google Chrome 24
- Safari 5

7.3 Wskazania dla twórców kolejnych aplikacji klienckich

Podczas dostosowywania aplikacji klienckiej dla nowego klienta trzeba pamiętać o następujących czynnościach:

1. Wprowadzenie identyfikatora aplikacji oraz jego klucza.
2. Aktualizacja bazy danych profili, albo poprzez usługę internetową, albo po przez odpowiednie skrypty napisane w języku *SQL*.

3. Stworzenie nowej kolejki, w której będą przesyłane wiadomości do części centralnej SCMS.
4. Odpowiednim skonfigurowaniu modułu komunikacyjnego po stronie klienta.
5. Stworzeniu nowego skryptu/nowej metody pozyskiwania pracowników, dla których mają zostać stworzone ewentualne nowe karty.
6. Wykonać drobne poprawki stylistyczne:
    - Zmienić logo firmy.
    - Zmienić nazwę firmy.
7. Instalacja profili dla danej aplikacji.
8. Określenie źródeł danych do personalizacji.

Dodatek A

Przypadki użycia

SCMS

Poniższe przypadki użycia dotyczą modułu administracyjnego. W niżej przedstawionych przypadkach użycia aktorem jest administrator systemu. Wyjątek stanowi PU-SCMS-1, w którym opisany jest proces uzyskania przez użytkownika roli administratora systemu.

PU-SCMS-1 - Logowanie do systemu
Aktor: Użytkownik
Priorytet: wysoki
Główny scenariusz:
1. Użytkownik podaje swój login, hasło.
2. Użytkownik loguje się do systemu.
Rozszerzenia:
2.A. Login lub hasło są niepoprawne.
2.A.1. System wyświetla odpowiedni komunikat.
2.A.2. Powrót do kroku **1**.
2.B. Użytkownik jest nieaktywny/zablokowany.
2.B.1. System wyświetla odpowiedni komunikat.
2.B.2. Powrót do kroku **1**.

Moduł administracyjny

PU-SCMS-ADM-1 / PU-SCMS-ADM-2 - Dodawanie nowej organizacji / edycja istniejącej organizacji
Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:
1. Administrator otwiera listę organizacji.
2. System prezentuje listę organizacji.
3. Administrator wybiera opcję dodania organizacji / edycji organizacji.
4. System prezentuje formularz.
5. Administrator podaje dane organizacji.

6. Administrator określa rolę organizacji (klient, dostawca kart, itd.)

7. Administrator potwierdza dane.

8. System zapisuje informacje.

Rozszerzenia:

2.A. Brak istniejących organizacji w systemie.

2.A.1. System wyświetla odpowiedni komunikat

2.A.2. Przejście do kroku **3**. / Koniec przypadku użycia.

7.A. Administrator podał niepoprawne dane.

7.A.1. System prezentuje odpowiednią informację o błędzie.

7.A.2. Powrót do kroku **5**.

PU-SCMS-ADM-3 - Usunięcie istniejącej organizacji

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator otwiera listę organizacji.

2. System prezentuje listę organizacji.

3. Administrator wybiera opcję usunięcia organizacji.

4. System prosi o potwierdzenie usunięcia.

5. Administrator potwierdza.

6. System usuwa organizację.

Rozszerzenia:

2.A. Brak istniejących organizacji w systemie.

2.A.1. System wyświetla odpowiedni komunikat

2.A.2. Koniec przypadku użycia.

5.A. Administrator nie potwierdził usunięcia organizacji.

5.A.1. Koniec przypadku użycia.

PU-SCMS-ADM-7 / PU-SCMS-ADM-8 - Dodanie nowej autoryzacji aplikacji klienckiej/ edycja istniejącej autoryzacji aplikacji klienckiej

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator wybiera akcję autoryzowania nowej aplikacji klienta / akcję modyfikacji istniejącej autoryzacji aplikacji klienta.

2. System wyświetla ekran pozwalający na autoryzację.

3. Administrator wybiera klienta, który otrzyma autoryzację.

4. Administrator wprowadza nazwę aplikacji klienta.

5. Administrator wprowadza datę wygaśnięcia autoryzacji.

6. Administrator wprowadza pulę adresów IP (zbiór adresów sieć + maska), dla których będzie ważna autoryzacja.

7. Administrator wprowadza klucz autoryzacji aplikacji klienckiej.

8. Administrator zatwierdza wprowadzone dane.

9. System autoryzuje aplikację klienta.

Rozszerzenia:

7.A. Administrator prosi o automatyczne wygenerowanie klucza.

7.A.1. System automatycznie uzupełnia pole wygenerowanym prawidłowym kluczem.

7.A.2. Przejście do kroku **8**.

8.A. Dowolny z wprowadzonych adresów jest niepoprawnym adresem IPv4 lub IPv6.

8.A.1. System wyświetla odpowiedni komunikat, ze wskazaniem na konkretne błędne adresy.

8.A.2. Powrót do kroku **6**.

8.B. Data wygaśnięcia autoryzacji jest nieprawidłowa (minęła)

8.B.1. System wyświetla odpowiedni komunikat.

8.B.2. Powrót do kroku **5**.

PU-SCMS-ADM-9 - Dezaktywacja autoryzacji aplikacji klienckiej

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator otwiera listę autoryzowanych aplikacji klientów.

2. System wyświetla listę autoryzowanych aplikacji klientów.

3. Administrator wybiera autoryzację aplikacji klienta do dezaktywowania.

4. Administrator wywołuje akcję dezaktywowania wybranej aplikacji klienta.

5. System prosi o potwierdzenie dezaktywacji.

6. Administrator potwierdza dezaktywację.

7. System dezaktywuje autoryzację klienta. // ustawia datę wygaśnięcia na obecną

Rozszerzenia:

2.A. Brak autoryzowanych klientów w systemie.

2.A.1. System wyświetla odpowiedni komunikat.

2.A.2. Koniec przypadku użycia.

6.A. Administrator nie potwierdza dezaktywacji.

6.A.1. System wyświetla odpowiedni komunikat.

6.A.2. Koniec przypadku użycia.

PU-SCMS-ADM-10 / PU-SCMS-ADM-11 - Dodanie/ Edycja roli w systemie SCMS

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator wybiera akcję dodania nowej roli / akcję modyfikacji istniejącej roli w systemie SCMS.

2. Administrator wybiera opcję dodania roli / edycji roli.

3. System prezentuje formularz.

4. Administrator podaje dane roli.

5. Administrator określa typ roli (rola dla organizacji lub rola w aplikacji)

6. Administrator potwierdza dane.

7. System zapisuje informacje.

Rozszerzenia:

2.A. Brak istniejących ról w systemie.
2.A.1. System wyświetla odpowiedni komunikat
2.A.2. Przejście do kroku **3**. / Koniec przypadku użycia.
7.A. Administrator podał niepoprawne dane.
7.A.1. System prezentuje odpowiednią informację o błędzie.
7.A.2. Powrót do kroku **5**.

PU-SCMS-ADM-12 - Usunięcie istniejącej roli
Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:
1. Administrator otwiera listę ról.
2. System prezentuje listę ról.
3. Administrator wybiera opcję usunięcia roli.
4. System prosi o potwierdzenie usunięcia.
5. Administrator potwierdza.
6. System usuwa rolę.
Rozszerzenia:
2.A. Brak istniejących ról w systemie.
2.A.1. System wyświetla odpowiedni komunikat
2.A.2. Przejście do kroku **3**. / Koniec przypadku użycia.
5.A. Administrator nie potwierdził usunięcia roli.
5.A.1. Koniec przypadku użycia.
6.A. Rola nie może być usunięta
6.A.1. System wyświetla odpowiedni komunikat
6.A.2. Koniec przypadku użycia

Moduł zarządzania kartami

PU-SMCS-ZK-1 / PU-SCMS-ZK-2 - Dodawanie nowego typu karty / edycja istniejącego typu karty
Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:
1. Administrator otwiera listę typów kart.
2. System prezentuje listę typów kart.
3. Administrator wybiera opcję dodania nowego typu karty / opcję edycji istniejącego typu karty.
4. System prezentuje formularz do wypełnienia.
5. Administrator wprowadza podstawowe dane o karcie.
6. Administrator wprowadza dodatkowe atrybuty karty.
7. Administrator zatwierdza formularz.
8. System zapamiętuje nowy typ karty / modyfikuje informacje o istniejącym typie karty.
Rozszerzenia:

2.A. Lista jest pusta.
2.A.1. Przejście do kroku **3**. / Koniec przypadku użycia.
7.A. Administrator nie podał wszystkich danych.
7.A.1. System wyświetla odpowiedni komunikat.
7.A.2. Powrót do kroku **5**.
7.B. Istnieją dwa atrybuty o takim samym identyfikatorze (nazwie)
7.B.1. System wyświetla odpowiedni komunikat.
7.B.2. Powrót do kroku **6**.
7.C. Administrator podał niepoprawne dane.
7.C.1. System wyświetla odpowiedni komunikat.
7.C.2. Powrót do kroku **5**.
7.D. Administrator podał nazwę typu karty, który już istnieje.
7.D.1. System wyświetla odpowiedni komunikat.
7.D.2. Powrót do kroku **5**.

PU-SMCS-ZK-3 - Usuwanie typu karty

Aktor: Administrator

Priorytet: średni

Główny scenariusz:

1. Administrator otwiera listę typów kart.
2. System prezentuje listę typów kart.
3. Administrator wybiera opcję usunięcia wskazanego typu karty.
4. System prosi o potwierdzenie.
5. Administrator potwierdza usunięcie.
6. System usuwa wskazany typ karty.

Rozszerzenia:

2.A. Lista typów kart jest pusta.
2.A.1. Koniec przypadku użycia.
5.A. Administrator nie potwierdza usunięcia.
5.A.1. Powrót do kroku **2**.

PU-SMCS-ZK-4 / PU-SMCS-ZK-5 - Dodawanie zestawu kluczy typu karty / edycja zestawu kluczy typu karty

Aktor: Administrator

Priorytet: średni

Główny scenariusz:

1. Administrator otwiera listę typów kart.
2. System prezentuje listę typów kart.
3. Administrator wybiera opcję zarządzania zbiorem kluczy wskazanego typu karty
4. System prezentuje zbiór kluczy wskazanego typu karty.
5. Administrator wybiera opcję dodania nowego zbioru kluczy / edycji istniejącego zbioru kluczy.
6. System prezentuje formularz.
7. Administrator wprowadza informacje o zbiorze kluczy.
8. Administrator podaje klucze.

9. Administrator potwierdza wprowadzone dane.

10. System zapisuje informacje.

Rozszerzenia:

2.A. Lista typów kart jest pusta.

2.A.1. Koniec przypadku użycia.

4.A. Zbiór kluczy dla danej karty jest pusty.

4.A.1. System wyświetla odpowiedni komunikat.

4.A.2. Przejście do kroku **5**. / Koniec przypadku użycia.

9.A. Wprowadzone dane są niepoprawne.

9.A.1. Powrót do kroku **7**.

PU-SMCS-ZK-6 - Usunięcie zestawu kluczy typu karty

Aktor: Administrator

Priorytet: średni

Główny scenariusz:

1. Administrator otwiera listę typów kart.

2. System prezentuje listę typów kart.

3. Administrator wybiera opcję zarządzania zbiorem kluczy wskazanego typu karty

4. System prezentuje zbiór kluczy wskazanego typu karty.

5. Administrator wybiera opcję usunięcia istniejącego zbioru kluczy.

6. System prosi o potwierdzenie.

7. Administrator potwierdza usunięcie.

8. System usuwa wskazany zbiór kluczy.

Rozszerzenia:

2.A. Lista typów kart jest pusta.

2.A.1. Koniec przypadku użycia.

4.A. Zbiór kluczy dla danej karty jest pusty.

4.A.1. System wyświetla odpowiedni komunikat.

4.A.2. Koniec przypadku użycia.

7.A. Administrator nie potwierdza usunięcia.

7.A.1. Koniec przypadku użycia.

PU-SMCS-ZK-7 / PU-SMCS-ZK-8 - Dodawanie aplikacji (na karty) / edycja aplikacji (na karty)

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator otwiera listę aplikacji (na karty).

2. System prezentuje listę aplikacji (na karty).

3. Administrator wybiera opcję dodania nowej aplikacji (na karty) / edycji istniejącej aplikacji (na karty).

4. System prezentuje formularz.

5. Administrator wprowadza podstawowe informacje o aplikacji (opis, wersja itd.).

6. Administrator wybiera, z którymi typami kart aplikacja jest kompatybilna.

7. Administrator definiuje, że aplikacja może zostać wprowadzona na kartę w pro-

cesie personalizacji.

8. Administrator dostarcza plugin instalujący aplikację w procesie personalizacji.

9. Administrator definiuje, w których etapach procesu personalizacji bierze udział plugin instalujący aplikację.

10. Administrator definiuje zbiór danych jaki plugin musi dostać podczas procesu personalizacji.

11. Administrator definiuje zbiór danych jaki plugin produkuje (udostępnia) w procesie personalizacji.

12. Administrator definiuje zbiór danych przechowywanych przez aplikację (dane, które aplikacja musi utrwalić pomiędzy zleceniami - np. numer kolejnego kodu kreskowego).

13. Administrator definiuje, że aplikacja może być zainstalowana już po procesie personalizacji.

14. Administrator definiuje operację instalacji aplikacji na spersonalizowanej karcie.

15. Administrator definiuje, że aplikacja może być odinstalowana już po procesie personalizacji.

16. Administrator definiuje operację deinstalacji aplikacji na spersonalizowanej karcie.

17. Administrator definiuje zbiór operacji, które mogą zostać wykonane podczas zarządzania aplikacją (self-service).

18. Administrator potwierdza wprowadzone dane.

19. System zapisuje informacje.

Rozszerzenia:

2.A. Lista aplikacji jest pusta.

2.A.1. Przejście do kroku **3**. / Koniec przypadku użycia.

7.A. Administrator nie potwierdza, że aplikacja może być instalowana w procesie personalizacji.

7.A.1. Przejście do kroku **13**.

13.A. Administrator nie potwierdza, że aplikacja może być instalowana po procesie personalizacji.

13.A.1. Przejście do kroku **15**.

15.A. Administrator nie potwierdza, że aplikacja może być odinstalowana po procesie personalizacji.

15.A.1. Przejście do kroku **17**.

18.A. Wprowadzone dane są niepoprawne (np. istnieje aplikacja o podanej nazwie i wersji).

18.A.1. Powrót do kroku **5**.

PU-SMCS-ZK-9 - Zablokowanie aplikacji (na karty)

Aktor: Administrator

Priorytet: średni

Główny scenariusz:

1. Administrator otwiera listę aplikacji (na karty).

2. System prezentuje listę aplikacji (na karty).

3. Administrator wybiera opcję zblokowania wskazanej aplikacji.

4. System pyta o potwierdzenie zablokowania aplikacji (na karty).

5. Administrator potwierdza zablokowanie.

6. System blokuje możliwość instalacji aplikacji na kartach i dodawania do profili.

Rozszerzenia:

2.A. Lista aplikacji jest pusta.

2.A.1. Koniec przypadku użycia.

4.A. Zbiór kluczy dla danej karty jest pusty.

4.A.1. System wyświetla odpowiedni komunikat.

4.A.2. Koniec przypadku użycia.

5.A. Administrator nie potwierdza zablokowania aplikacji (na karty).

5.A.1. Koniec przypadku użycia.

PU-SCMS-ZK-10 / PU-SCMS-ZK-11 - Dodawanie profilu karty / edycja profilu karty

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator otwiera listę profili kart.

2. System prezentuje listę profili kart.

3. Administrator wybiera opcję dodania nowego profilu karty / edycji istniejącego profilu karty.

4. System prezentuje formularz.

5. Administrator określa nazwę profilu.

6. Administrator wybiera aplikacje klienckie, które mogą korzystać z danego profilu karty.

7. Administrator określa które aplikacje (na karty) muszą zostać zainstalowane w procesie personalizacji.

8. Administrator wybiera aplikacje (na karty), które będą mogły być opcjonalnie zainstalowane w procesie personalizacji.

9. System szereguje wybrane aplikacje (na karty) - w jakiej kolejności zostaną zainstalowane w procesie personalizacji karty z danym profilem.

10. Administrator sprawdza poprawność uszeregowania aplikacji (na karty).

11. Administrator określa które aplikacje (na karty) będą mogły być zarządzane przez właściciela karty (self-service).

12. Administrator określa dla wybranych w kroku 11 aplikacji (na karty) operacje, które właściciele kart będą mogli wykonać samodzielnie (self-service).

13. Administrator potwierdza wprowadzone dane.

14. System zapisuje informacje.

Rozszerzenia:

2.A. Lista profili kart jest pusta.

2.A.1. Przejście do kroku **3**. / Koniec przypadku użycia.

10.A. Uszeregowanie aplikacji jest niepoprawne.

10.A.1. Administrator ręcznie poprawia uszeregowanie aplikacji.

10.A.2. Przejście do kroku **11**.

13.A. Wprowadzone dane są niepoprawne.
13.A.1. Powrót do kroku **7**.

PU-SCMS-ZK-12 - Zablokowanie korzystania z profilu kart
Aktor: Administrator
Priorytet: średni
Główny scenariusz:
1. Administrator otwiera listę profili kart.
2. System prezentuje listę profili kart.
3. Administrator wybiera opcję zablokowania profilu kart.
4. System prosi o potwierdzenie.
5. Administrator potwierdza zablokowania.
6. System blokuje wykorzystanie profilu do kolejnych personalizacji.
Rozszerzenia:
2.A. Lista profili kart jest pusta.
2.A.1. Koniec przypadku użycia.
4.A. Administrator nie potwierdza usunięcia.
4.A.1. Koniec przypadku użycia.

PU-SCMS-ZK-13 - Usuwanie kodów ATR
Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:
1. Administrator otwiera listę modeli blankietow.
2. System prezentuje listę modeli blankietow.
3. Administrator wybiera opcję wyswietlenia kodów ATR blankietów.
4. System prezentuje formularz z kodami ATR.
5. Administrator wybiera opcję usunięcia kodu ATR
6. System prosi o potwierdzenie
Rozszerzenia:
2.A. Lista kodów modeli blankietów jest pusta.
2.A.1. Koniec przypadku użycia.
4.A. Lista kodów kodów ATR modelu blankietu jest pusta.
4.A.1. Koniec przypadku użycia.
7.A. Administrator nie potwierdza usunięcia kodu.
7.A.1. Koniec przypadku użycia.

PU-SCMS-ZK-14 / PU-SCMS-ZK-15 - Dodawanie/Edycja kodów ATR
Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:
1. Administrator otwiera listę modeli blankietów.
2. System prezentuje listę modeli blankietów.
3. Administrator wybiera opcję wyświetlenia kodów ATR blankietów.
4. System prezentuje formularz z kodami ATR.

5. Administrator wybiera opcję dodania/edycji kodów ATR
6. System prezentuje formularz
7. Administrator wprowadza dane kodu ATR
8. Administrator potwierdza wprowadzone dane.
9. System zapisuje informacje.

Rozszerzenia:

2.A. Lista kodów modeli blankietów jest pusta.
2.A.1. Koniec przypadku użycia.
4.A. Lista kodów kodów ATR modelu blankietu jest pusta.
4.A.1. Koniec przypadku użycia / przejście do kroku **5**.
7.A. Wprowadzone dane są niepoprawne.
7.A.1. Powrót do kroku **6**.

PU-SCMS-ZK-16 - Usunięcie metody dywersyfikacji

Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:

1. Administrator otwiera listę metod.
2. System prezentuje listę metod.
3. Administrator wybiera opcję usunięcia metody.
4. System prosi o potwierdzenie usunięcia.
5. Administrator potwierdza.
6. System usuwa metodę.

Rozszerzenia:

2.A. Brak istniejących metod w systemie.
2.A.1. System wyświetla odpowiedni komunikat
2.A.2. Przejście do kroku **3**. / Koniec przypadku użycia.
5.A. Administrator nie potwierdził usunięcia metody.
5.A.1. Koniec przypadku użycia.

PU-SCMS-ZK-17 / PU-SCMS-ZK-18 - Dodawanie metody dywersyfikacji / edycja metody dywersyfikacji

Aktor: Administrator
Priorytet: wysoki
Główny scenariusz:

1. Administrator otwiera listę metod.
2. System prezentuje listę metod.
3. Administrator wybiera opcję dodania nowej metody / edycji istniejącej metody.
4. System prezentuje formularz.
5. Administrator wprowadza dane metody dywersyfikacji
6. Administrator potwierdza wprowadzone dane.
7. System zapisuje informacje.

Rozszerzenia:

2.A. Lista metod jest pusta.
2.A.1. Przejście do kroku **3**. / Koniec przypadku użycia.

6.A. Wprowadzone dane są niepoprawne.

6.A.1. Powrót do kroku **5**.

PU-SCMS-ZK-17 / PU-SCMS-ZK-18 - Dodawanie metody dywersyfikacji / edycja metody dywersyfikacji

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Administrator otwiera listę metod.

2. System prezentuje listę metod.

3. Administrator wybiera opcję dodania nowej metody / edycji istniejącej metody.

4. System prezentuje formularz.

5. Administrator wprowadza dane metody dywersyfikacji

6. Administrator potwierdza wprowadzone dane.

7. System zapisuje informacje.

Rozszerzenia:

2.A. Lista metod jest pusta.

2.A.1. Przejście do kroku **3**. / Koniec przypadku użycia.

6.A. Wprowadzone dane są niepoprawne.

6.A.1. Powrót do kroku **5**.

PU-SCMS-ZK-19 / PU-SCMS-ZK-20 - Dodawanie informacji o pluginie personalizacji / Edycja informacji o pluginie personalizacji

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. System prezentuje listę pluginów.

2. Administrator wybiera opcję dodania pluginu / edycji pluginu.

3. System prezentuje formularz.

4. Administrator podaje dane o pluginie (patrz: schmat bazy danych).

5. Administrator potwierdza dane.

6. System zapisuje informacje.

Rozszerzenia:

2.A. Brak istniejących pluginów w systemie.

2.A.1. System wyświetla odpowiedni komunikat

2.A.2. Przejście do kroku **3**. / Koniec przypadku użycia.

6.A. Administrator podał niepoprawne dane.

6.A.1. System prezentuje odpowiednią informację o błędzie (miej na uwadze: niespójność bazy danych, typy danych).

6.A.2. Powrót do kroku **3**.

PU-SCMS-ZK-21 - Usunięcie informacji o pluginie personalizacji

Aktor: Administrator

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera listę pluginów.
2. System prezentuje listę pluginów.
3. Użytkownik wybiera opcję usunięcia pluginu.
4. System pyta o potwierdzenie.
5. Użytkownik potwierdza dane.
6. System zapisuje informacje.

Rozszerzenia:

2.A. Brak istniejących pluginów w systemie.
2.A.1. System wyświetla odpowiedni komunikat
2.A.2. Koniec przypadku użycia
4.A. Nie można usunąć informacji o pluginie ze względu na fakt wykorzystywania go w jakimś profilu karty lub w innej relacji.
4.A.1. System prezentuje odpowiedni komunikat ze wskazaniem na problematyczne profile i powiązania.
4.A.2. Koniec przypadku użycia.
5.A. Użytkownik nie potwierdził danych.
5.A.1. Koniec przypadku użycia.

Aplikacja kliencka

Poniżej przedstawione przypadki użycia dotyczą przykładowego modułu aplikacji klienckiej. W poniższych przypadkach użycia, pod pojęciem aktora Użytkownik - rozumie się operatora aplikacji klienckiej, uprzywilejowanego do obsługi aplikacji.

PU-AK-1 - Logowanie do systemu
Aktor: Użytkownik
Priorytet: wysoki
Główny scenariusz:
1. Użytkownik otwiera aplikację.
2. System prosi o podanie loginu i hasła.
3. Użytkownik podaje swój login i hasło.
4. System autoryzuje użytkownika.

Rozszerzenia:

4.A. Login lub hasło są niepoprawne.
4.A.1. System wyświetla odpowiedni komunikat.
4.A.2. Powrót do kroku **2**.
4.B. Użytkownik jest nieaktywny/zablokowany.
4.B.1. System wyświetla odpowiedni komunikat.
4.B.2. Powrót do kroku **2**.

Moduł zleceń

PU-AK-ZL-1 - Przegląd kart (aktywnych, zleconych i wycofanych)
Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera widok przeglądania kart.

2. System prezentuje listę kart.

3. Użytkownik przegląda listę, filtruje ją i sortuje.

4. Użytkownik wybiera kartę z listy.

5. System prezentuje dane szczegółowe dotyczące wybranej karty.

PU-AK-ZL-2 / PU-AK-ZL-3 - Zlecenie wydania nowej karty / edycja zlecenia wydania nowej karty.

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera listę pracowników

2. System prezentuje listę pracowników.

3. Użytkownik filtruje pracowników, dla których mają być wygenerowane nowe karty.

4. Użytkownik potwierdza wybór pracowników.

5. System prosi o wybór profilu karty.

6. Użytkownik wybiera profil.

7. System przyjmuje zlecenie do realizacji.

Rozszerzenia:

6.A. Brak profili karty

6.A.1. System wyświetla komunikat błędu.

6.A.2. Koniec przypadku użycia

PU-AK-ZL-4 - Podgląd karty

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik przegląda karty (PU-AK-ZL-1).

2. Użytkownik wybiera kartę z listy.

3. System prezentuje dane szczegółowe dotyczące wybranej karty.

PU-AK-ZL-5 - Usunięcie zlecenia wydania karty

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera widok zleceń.

2. System prezentuje widok zleceń.

3. Użytkownik wywołuje akcję usunięcia wskazanego zlecenia.

4. System prosi o potwierdzenie.

5. Użytkownik potwierdza akcję.

6. System usuwa zlecenia.

PU-AK-ZL-6 - Zlecenie wydania duplikatu karty

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera listę kart.
2. Użytkownik wywołuje akcję wydania duplikatu dla wskazanej karty.
3. System prezentuje nazwę profilu karty.
4. System prosi o potwierdzenie.
5. Użytkownik potwierdza akcję.
6. System generuje zlecenie wydania duplikatu.

PU-AK-ZL-7 - Zablokowanie karty

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera listę kart.
2. Użytkownik wywołuje akcję zablokowania dla wskazanej karty.
3. System prosi o potwierdzenie.
4. Użytkownik potwierdza akcję.
5. System blokuje kartę.

PU-AK-ZL-8 - Odblokowanie karty

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera listę kart.
2. Użytkownik wywołuje akcję odblokowania dla wskazanej karty.
3. System prosi o potwierdzenie.
4. Użytkownik potwierdza akcję.
5. System odblokowuje kartę.

Moduł administracyjny z modułem raportowania

PU-AK-ADM-1 - Modyfikacja klucza autoryzacyjnego aplikacji klienckiej

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik wybiera ustawienia administracyjne.
2. System prezentuje ustawienia aplikacji.
3. Użytkownik wprowadza klucz autoryzacji aplikacji klienckiej.
4. Użytkownik potwierdza wprowadzone zmiany.
5. System zapisuje zmiany.

Rozszerzenia:

5.A. Nie jest możliwa weryfikacja kodu autoryzacyjnego.

5.A.1. System wyświetla odpowiednią informację.

5.A.2. Powrót do kroku 3.
5.B. Kod autoryzacji jest nieprawidłowy.
5.B.1. System wyświetla odpowiednią informację.
5.B.2. Powrót do kroku 3.
5.C. Wprowadzony kod autoryzacji wygasł.
5.C.1. System wyświetla odpowiednią informację.
5.C.2. Powrót do kroku 3.

Moduł profilu

PU-AK-PF-1 - Synchronizacja listy pracowników
Aktor: Użytkownik
Priorytet: wysoki
Główny scenariusz:
1. Użytkownik wybiera opcję synchronizacji listy pracowników.
2. System odczytuje dane z systemu zewnętrznego.
3. System wyświetla informacje o liczbie pobranych rekordów pracowników.
Rozszerzenia:
2.A. Błąd synchronizacji.
2.A.1. System wyświetla komunikat.
2.A.2. Koniec przypadku użycia.

PU-AK-PF-2 / PU-AK-PF-3 - Dodanie zdjęcia Użytkownika / zmiana zdjęcia pracownika
Aktor: Użytkownik
Priorytet: wysoki
Główny scenariusz:
1. Użytkownik otwiera listę pracowników.
2. System prezentuje listę pracowników.
3. Użytkownik wybiera pracownika, dla którego chce dodać/zmienić zdjęcie.
4. System prezentuje dane pracownika.
5. Użytkownik wywołuje akcję dodania/zmiany zdjęcia pracownika.
6. System prosi o wybranie zdjęcia z komputera Użytkownika.
7. Użytkownik wybiera zdjęcie ze swojego komputera.
8. System zapisuje zdjęcie przypisane do użytkownika.
Rozszerzenia:
8.A. Nierozpoznawalny format pliku.
8.A.1. System wyświetla komunikat.
8.A.2. Powrót do kroku 6.

PU-AK-PF-4 - Dodanie wartości do słownika pojęć / edycja wartości ze słownika
Aktor: Użytkownik
Priorytet: wysoki
Główny scenariusz:
1. Użytkownik otwiera słownik pojęć.

2. System prezentuje słownik pojęć.
3. Użytkownik wywołuje akcję dodania nowego/edycji pojęcia.
4. System prezentuje formularz do wypełnienia.
5. Użytkownik wprowadza nazwę pojęcia.
6. Użytkownik wprowadza opis pojęcia.
7. Użytkownik zatwierdza formularz.
8. System zapamiętuje nowe pojęcie.

PU-AK-PF-5 - Usunięcie wartości ze słownika

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera słownik pojęć.
2. System prezentuje słownik pojęć.
3. Użytkownik wywołuje akcję usunięcia pojęcia.
4. System prosi o potwierdzenie.
5. Użytkownik potwierdza akcję.
6. System usuwa pojęcie.

PU-AK-PF-6 - Przegląd listy pracowników

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik otwiera listę pracowników.
2. System prezentuje listę pracowników.
3. Użytkownik przegląda listę, filtruje ją i sortuje.
4. Użytkownik wybiera pracownika z listy.
5. System prezentuje dane szczegółowe dotyczące wybranego pracownika.

PU-AK-PF-7 - Przegląd kart przypisanych do danego pracownika

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik przegląda listę pracowników (PU-AK-PF-6).
2. Użytkownik wywołuje akcję wyświetlenia kart pracownika.
3. System prezentuje karty pracownika.

PU-AK-PF-8 - Instalacja/aktualizacja profilu karty

Aktor: Użytkownik

Priorytet: wysoki

Główny scenariusz:

1. Użytkownik wybiera opcję instalacji profilu karty.
2. System prezentuje dostępne dla aplikacji profile.
3. Użytkownik zaznacza profile, które mają zostać zainstalowane/uaktualnione.
4. System prosi o potwierdzenie.

5. Użytkownik potwierdza operację.
6. System pobiera dane profilu z usługi sieciowej.
7. Koniec przypadku użycia.

Rozszerzenia:

5.A. Użytkownik nie potwierdza operacji.
5.A.1. Koniec przypadku użycia.
6.A. System napotkał błąd podczas pobierania danych.
6.A.1. System informuje użytkownika o błędzie.
6.A.2. Powrót do pkt 2.

Moduł self-service

PU-AK-SS-1 - Zablokowanie karty

Aktor: Użytkownik
Priorytet: wysoki
Główny scenariusz:

1. Posiadacz karty loguje się do systemu.
2. Posiadacz karty wywołuje opcję zablokowania karty.
3. System prosi o potwierdzenie.
4. Posiadacz karty potwierdza akcję.
5. System blokuje kartę.

Dodatek B

Płyta CD

- Dokumentacja protokołu komunikacyjnego
- Dokumentacja usługi sieciowej
- Instrukcja zarządzania systemem informatycznym
- Instrukcja_rabbitmq
- Polityka bezpieczeństwa
- Przekazanie do eksploatacji
- SCMS_CLIENT_schemat_bazy_danych
- SCMS_schemat_bazy_danych

Bibliografia

[1] http://computer.howstuffworks.com/question332.htm

[2] http://www.phpdoc.org/docs/latest/index.html

[3] http://www.phpdoc.org/docs/latest/for-users/phpdoc/tags/see.html

[4] http://students.mimuw.edu.pl/SO/Projekt02-03/temat4-g2/miroslaw_szymanski/sc.htm

[5] strona projektu biblioteki https://github.com/videlalvaro/php-amqplib

[6] przypis, więcej informacji o aplikacji komunikacyjnej znajduje się w załączniku Instrukcja obsługi RabbitMQ dla administratora oraz na stronie projektu http://rabbitmq.com

[7] Standardy opracowane przez Politechnikę Poznańską http://conaiten.put.poznan.pl/projects/templates/wiki/Standard_kodowania_w_języku_PHP

[8] https://conaiten.put.poznan.pl/attachments/download/2470/PEKAtry.png

[9] Szczegółowy opis interfejsu do tego systemu znajduje się w dokumencie http://webservices.put.poznan.pl/eKadryService-current.wsdl

[10] Opis usługi dostępny jest tutaj: http://webservices.put.poznan.pl/eKontoService2.wsdl

[11] Przykłady obrazujące działanie komponentu: http://www.datatables.net/examples/