Łukasz Miedziński, Mateusz Smoliński
Instytut Informatyki, Politechnika Łódzka
Wólczańska 215, 90-924 Łódź
email: lukasz@miedzinski.org, mateusz.smolinski@p.lodz.pl

SYSTEM GRUPOWEGO ZAMAWIANIA POSIŁKÓW DLA PRACOWNIKÓW PRZEDSIĘBIORSTWA

Streszczenie - Współcześnie rynek pracy jest zorientowany na efektywność i wysoką jakość świadczonych usług. Synergia tych czynników jest widoczna w różnych obszarach i wpływa na organizację współczesnych miejsc pracy, co w szczególności jest zauważalne w korporacjach. Optymalizacje i usprawnienia organizacyjne modelu pracy przekładają się na obniżenie kosztów i zwiększenie konkurencyjności. Jednym z powtarzających się problemów wśród pracodawców dysponujących dużym zespołem pracowników jest codzienne zamawianie posiłków przez pracowników od zewnętrznych dostawców. Nieefektywny przepływ informacji pomiędzy pracownikami, pracodawcą a dostawcami posiłków ma wpływ na obniżenie efektywności pracy. Problem jest jeszcze bardziej dotkliwy dla pracodawcy, który oferuje możliwość częściowego lub całkowitego dofinansowania posiłków dla pracowników, ponieważ trzeba uwzględnić jeszcze kwestie rozliczeń finansowych. Autorzy przedstawili rozwiązanie wspomnianego problemu poprzez stworzenie autorskiego systemu informatycznego wspomagającego grupowe zamawianie posiłków przez pracowników przedsiębiorstwa.

Słowa kluczowe: Java Enterprise Edition, AngularJS, grupowe zamawianie posiłków

1 Wprowadzenie

W dużych korporacjach popularne jest indywidualne zamawianie posiłków przez pracowników, zamówione w ten sposób posiłki w stosunkowo krótkim czasie są dostarczane do siedziby firmy. Jest to rozwiązanie korzystne dla pracowników w sytuacji, gdy w siedzibie firmy brakuje punktów gastronomicznych, a pracownik w trakcie krótkiej przerwy w pracy chce się posilić. Koordynowanie wielu zamówień posiłków bez narzędzi wspomagających wymaga dodatkowego nakładu czasu pracowników, co nie przekłada się bezpośrednio na ich produktywność. Dodatkowo przy zamówieniach skierowanych do różnych dostawców trudno zapewnić w podobnym czasie dostarczenie wszystkich posiłków, co dodatkowo dezorganizuje przerwy w pracy zespołowej. W sytuacji zamówień grupowych problem jest jeszcze bardziej złożony, niż w przypadku zamówień indywidualnych. Wynika to z faktu, że potrzebne jest ustalenie szczegółów zamówienia grupowego

poprzez pozyskanie informacji od poszczególnych pracowników co do wybranych przez nich posiłków, a następnie należy rozliczyć koszty posiłków jak i ich dostawy do siedziby firmy.

Stosowanie grupowego zamawiania posiłków pozwala na zmniejszenie kosztów ale również powoduje obniżenie efektywności pracy. Stosowanym popularnie rozwiązaniem jest wyznaczanie osoby odpowiedzialnej za złożenie zamówień posiłków w danym dniu roboczym, W praktyce oznacza to przekazanie wymienionych trudności wskazanej osobie, która ponosi odpowiedzialność za zebranie zamówień od współpracowników oraz rozliczenie kosztów pomiędzy zamawiającymi. Należy podkreślić, że przekazanie odpowiedzialności wskazanemu pracownikowi jest jedynie rozwiązaniem ograniczającym obniżenie efektywności pracy. Bez dedykowanych narzędzi wspomagających zamawianie posiłków dla pracowników firmy, kwestie koordynowania wyboru posiłku i czasu dostawy oraz rozliczeń kosztów będą nadal wymagały dodatkowego nakładu czasu.

Naturalnym rozwiązaniem przedstawionego problemu jest stworzenie dedykowanego systemu informatycznego, który usprawni wymianę informacji pomiędzy pracownikami, dostawcami posiłków oraz pracodawcami przy codziennym zamawianiu posiłków dla grup pracowników w przedsiębiorstwie. Oprócz wspomagania procesu kompletowania grupowego zamówienia posiłków, koordynowania zamówień złożonych u różnych dostawców zaproponowany system ułatwi też rozliczenie kosztów tych zamówień. Stworzony system został przygotowany tak, by obsługiwał sytuacje kiedy pracownicy samodzielnie pokrywają koszty zamówionych posiłków jak i sytuacje, gdy pracodawca częściowo lub w całości dofinansowuje koszty takich zamówień. W kolejnych rozdziałach zaprezentowano autorski wielodostępny system "Let's Eat", składający się z aplikacji internetowej i relacyjnej bazy danych, który wspomaga grupowe zamawianie posiłków dla pracowników przedsiębiorstwa.

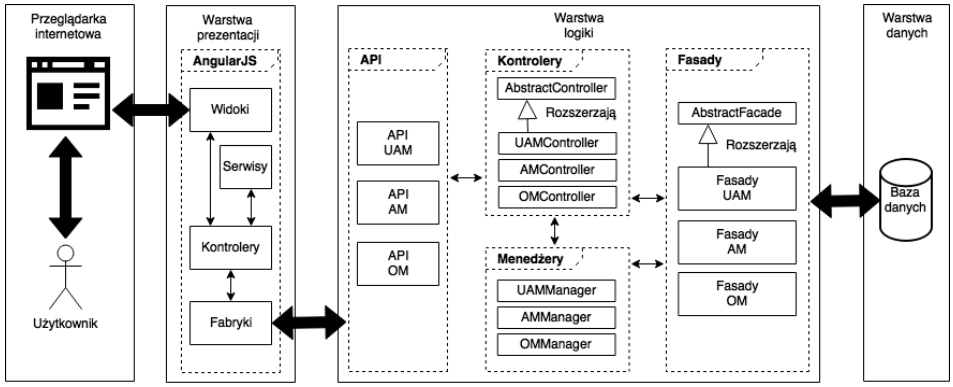
2 Założenia projektowe

Wszystkie założenia jakie budowany system powinien spełniać zostały ustalone z przedstawicielami łódzkiego oddziału firmy Sii, która zgłosiła problem braku odpowiednich narzędzi wspomagających grupowe zamawianie posiłków. Pierwszym z nich było użycie nowoczesnych technologii do budowy systemu. W projekcie systemu zdecydowano się więc na architekturę warstwową, co pozwoliło wprowadzić podział odpowiedzialności na poszczególne warstwy oprogramowania. Wyróżniono:

- warstwę danych, która obejmuje system zarządzania bazami danych PostgreSQL z relacyjną bazą danych [3],

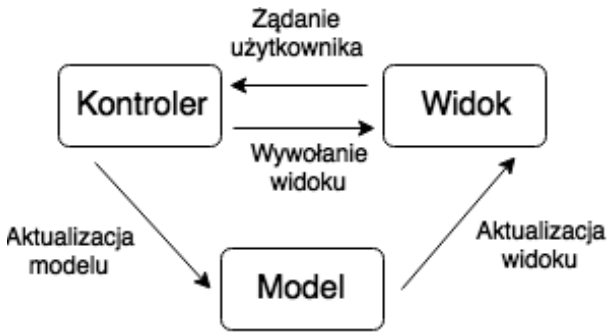
- warstwę logiki, która obejmuje aplikację internetową w technologii Java Enterprise Editon obsługującą logikę biznesową systemu rozumianą jako reguły przetwarzania danych biznesowych [7][10],
- warstwę prezentacji, która obejmuje aplikację w technologii AngularJS zapewniająca interfejs użytkownika osadzony w stronach WWW [4].

W doborze technologii uwzględniono ich wpływ na wydajność tworzonego systemu [4][5]. Technologie zastosowane w poszczególnych warstwach systemu przedstawia diagram na rysunku Rys. 1 Należy zaznaczyć, że zastosowano odmienne technologie w komunikacji pomiędzy warstwą prezentacji i logiki oraz między warstwą logiki i warstwą danych.



Rys. 1. Przekrój technologiczny wielowarstwowego systemu

Warstwa danych jest całkowicie odizolowanym elementem systemu, jednakże zarówno aplikacja warstwy logiki, jak i aplikacja warstwy prezentacji działają w serwerze aplikacyjnym Payara uruchomionym w kontenerze Docker [6]. Komunikacja pomiędzy warstwą logiki i warstwą prezentacji odbywa się poprzez interfejs API zbudowany w technologii REST (REpresentational State Transfer). Wykorzystane zostały do tego zapytania HTTP oraz ich najpopularniejsze metody, czyli: GET, POST, PUT i DELETE. Aplikacja warstwy logiki oraz aplikacja warstwy prezentacji została zaimplementowana zgodnie ze wzorcem architektonicznym Model-Widok-Kontroler w wersji 2 (Model-View-Controller) [8] (Rys. 2).



Rys. 2. Schemat przedstawiający wzorzec Model-Widok-Kontroler

Kolejnym istotnym aspektem jest zapewnienie wielodostępności budowanego systemu, czyli możliwości jego obsługi przez wielu użytkowników jednocześnie. Dla każdego użytkownika system musi zapewniać konto z przypisanymi poziomami dostępu, których zadaniem jest grupowanie uprawnień ze względu na rolę, jaką dany użytkownik ma pełnić w systemie. Dla pojedynczego konta system obsługuje dowolną kombinację poziomów dostępu przypisanych do konta. W projektowanym systemie wyróżniono następujące poziomy poziomu.

- Gość (Guest) - poziom dostępu grupujący przypadki użycia, które nie wymagają uwierzytelnienia w aplikacji.
- Administrator (Administrator) - poziom dostępu pozwalający na zarządzanie systemem i użytkownikami.
- Klient (Client) - poziom dostępu pozwalający na składanie zamówień w systemie.
- Menadżer (Manager) - poziom dostępu umożliwiający zamawianie i odbiór posiłków od dostawców.
- Księgowy (Accountant) - poziom dostępu pozwalający na przeglądanie raportów finansowych z określonego przedziału czasu dotyczących dostawcy posiłków lub pracownika przedsiębiorstwa.

Należy zauważyć, że w tworzonym systemie uwierzytelnianie użytkowników będzie realizowane z wykorzystaniem loginu i hasła, a wzorce do uwierzytelniania będą przechowywane w relacyjnej bazie danych w postaci niejawnej.

W celu zapewniania elastyczności tworzona aplikacja internetowa została podzielona na moduły funkcjonalne, które grupują przypadki użycia operujące na tych samych zbiorach danych. Podział na moduły funkcjonalne umożliwia uruchomienie systemu z ograniczoną funkcjonalnością, przy wyborze jedynie części dostępnych modułów funkcjonalnych.

W systemie wyróżnione zostały następujące moduły funkcjonalne.

- Moduł obsługi kont ozn. UAM (User Account Module)-funkcjonalność związana z kontami użytkowników. Przypadki użycia oferowane przez ten moduł przedstawia tabela Tab. 1.
- Moduł administracji ozn. AM (Administration Module)-funkcjonalność związana z ustawieniami i zarządzaniem systemem. Przypadki użycia oferowane przez ten moduł przedstawia tabela Tab. 2.
- Moduł zamówień ozn. OM (Order Module)-funkcjonalność związana z zamówieniami posiłków. Przypadki użycia oferowane przez ten moduł przedstawia tabela Tab. 3.

Tabela. 1. Macierz decyzyjna przypadków użycia dla modułu funkcjonalnego UAM

Lp.	Przypadek użycia	Goś.	Nie.	Adm.	Kli.	Men.	Ksi.
1	Zarejestruj	X					
2	Utwórz konto			X			
3	Zablokuj konto			X			
4	Odblokuj konto			X			
5	Edytuj poziomy dostępu			X			
6	Zmień hasło konta			X			
7	Edytuj dane konta			X			
8	Przeglądaj listę kont			X			
9	Edytuj dane własnego konta			X	X	X	X
10	Zmień własne hasło			X	X	X	X
11	Zaloguj	X					
12	Wyloguj		X	X	X	X	X
13	Zresetuj hasło	X					

Tabela. 2. Macierz decyzyjna przypadków użycia dla modułu funkcjonalnego AM

Lp.	Przypadek użycia	Goś.	Nie.	Adm.	Kli.	Men.	Ksi.
1	Wyświetl listę kategorii			X			
2	Dodaj kategorię			X			
3	Edytuj kategorię			X			
4	Wyświetl listę dostawców			X			
5	Dodaj dostawcę			X			
6	Edytuj dostawcę			X			
7	Wyświetl listę posiłków			X			
8	Dodaj posiłek			X			
9	Edytuj posiłek			X			
10	Usuń posiłek			X			
11	Wyświetl listę ustawień systemu			X			
12	Edytuj ustawienie systemu			X			

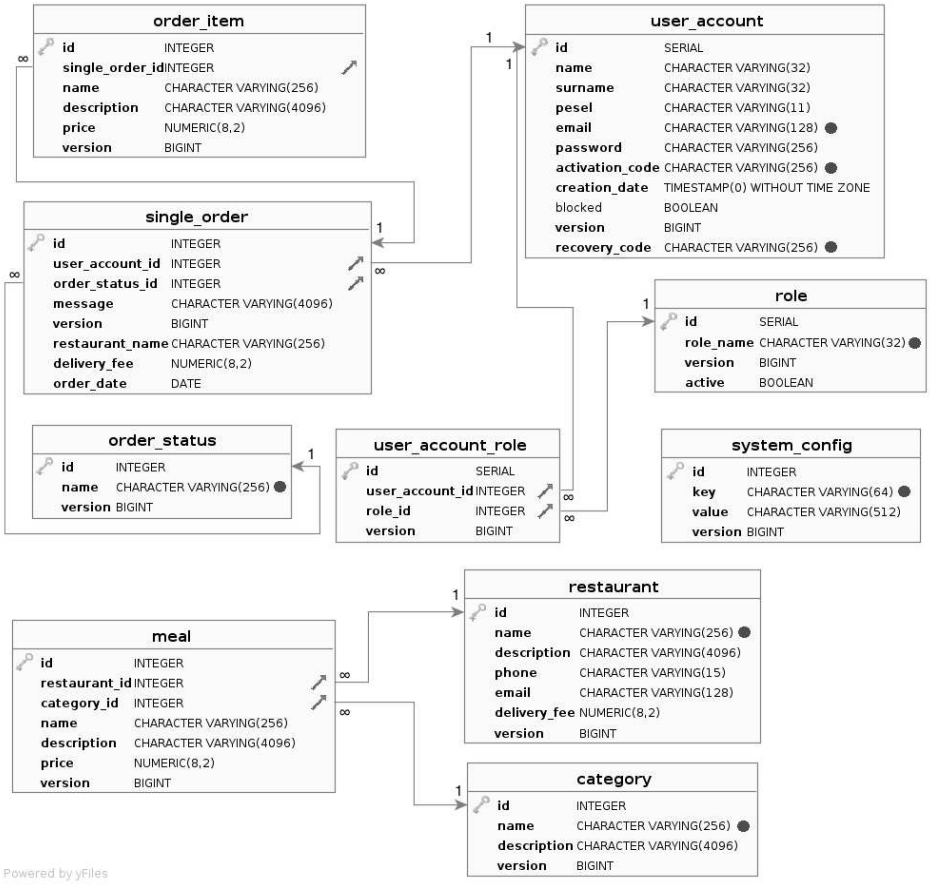
W wyniku analizy przedstawionych w tabelach przypadków użycia opracowano model danych i interfejs użytkownika tworzonego systemu. Model danych przedstawiono w postaci struktur relacyjnej bazy danych, która przechowuje dane przetwarzane przez system. Natomiast interfejs użytkownika w tworzonym systemie jest dostarczany przez strony WWW. Uzyskanie dostępu do interfejsu użytkownika w stworzonym systemie wymaga posiadania przeglądarki internetowej wspierającej technologię Java Script, współcześnie wymóg ten spełnia większość popularnych przeglądarek internetowych. Zastosowanie Java Script umożliwia odciążenie serwera poprzez przeniesienie do przeglądarki klienta generowania i obsługi interfejsu użytkownika systemu "Let's Eat".

Tabela. 3. Macierz decyzyjna przypadków użycia dla modułu funkcjonalnego OM

Lp.	Przypadek użycia	Goś.	Nie.	Adm.	Kli.	Men.	Ksi.
1	Wyświetl listę dostawców				X		
2	Wyświetl listę posiłków dla dostawcy				X		
3	Wybierz dostawcę do zamówienia				X		
4	Zmień dostawcę do zamówienia				X		
5	Dodaj posiłek do zamówienia				X		
6	Usuń posiłek z zamówienia				X		
7	Wyświetl szczegóły zamówienia				X		
8	Wyświetl listę swoich poprzednich zamówień				X		
9	Wyświetl szczegóły swojego poprzedniego zamówienia				X		
10	Wyświetl listę dzisiejszych zamówień					X	
11	Zmień statusy dzisiejszych zamówień					X	
12	Wyświetl raport wydatków z podziałem na dostawców z wybranego zakresu czasu						X
13	Wyświetl raport wydatków na poszczególnych użytkowników z wybranego zakresu czasu						X

3 Model relacyjnej bazy danych

Warstwa danych systemu zrealizowana została przy użyciu relacyjnej bazy danych obsługiwanej przez system zarządzania bazami danych PostgreSQL, uruchomiony w osobnym serwerze. Rys. 3 przedstawia diagram struktur bazy danych stworzonej dla systemu "Let's Eat", na diagramie wprowadzono dodatkowe oznaczenia. Symbolem złotego klucza oznaczone zostały klucze główne poszczególnych tabel w relacyjnej bazie danych. Czarne i niebieskie strzałki przedstawiają relacje, reprezentowane przez klucze obce. Pola z ograniczeniem unikalności wartości w zakresie kolumny tabeli oznaczono czerwoną kropką.



Rys. 3. Diagram przedstawiający struktury relacyjnej bazy danych

Dodatkowo wymuszono brak pustych pól w przechowywanych w bazie rekordach. W relacyjnym modelu danych ograniczono również reprezentacje kwot pieniężnych do wartości nieujemnych. Przyjęto także

minimalną długość łańcuchów tekstowych w bazie danych równą trzem znakom - nie mogą one również przekroczyć długości maksymalnej definiowanej poprzez typ pola. W celu czytelnego rozdzielenia uprawnień dostępu do poszczególnych tabel, dla każdego modułu funkcjonalnego utworzono osobne konto bazodanowe.

4 Implementacja reguł przetwarzania danych

W tworzonym systemie do implementacji reguł przetwarzania danych zastosowano referencyjne komponenty EJB (Enterprise Java Beans). Przykładem zastosowania bezstanowych komponentów EJB są fasady, odpowiedzialne za dostarczenie metod do realizacji operacji na danych przechowywanych w relacyjnej bazie danych. Ze względu na korzystanie przez moduły funkcjonalne z różnych kont bazodanowych, klasy fasad są zapewniane dla każdego z nich. Część wspólną implementacji wszystkich fasad zawiera bazowa klasa abstrakcyjna, która udostępnia metody realizujące podstawowe operacje na danych m.in. odczyt, dodawanie, usuwanie i edycję. Komponenty fasad wykonywanie operacji na danych delegują do menadżera encji. Oprócz komponentów fasad zastosowano dwa inne rodzaje komponentów EJB, m. in. bezstanowe menadżery oraz stanowe kontrolery. Najprostsze implementacje przypadków użycia nie stosują komponentów menadżerów EJB.

Zastosowanie odwzorowania obiektowo-relacyjnego jest możliwe poprzez wymienione wcześniej obiekty encji zgodne ze standardem JPA (Java Persistence API), gdzie każdy obiekt encji reprezentuje rekord z relacyjnej bazy danych [8][9]. Relacje pomiędzy rekordami z różnych tabel w bazie danych są reprezentowane jako zależności pomiędzy obiektami encji. Natomiast klasy encji odwzorowują poszczególne tabele bazodanowe. W implementacji encji zastosowano także dodatkowe ograniczenia w standardzie Bean Validation dla reprezentowanych przez nie danych. Zapewniło to niezależną od modelu relacyjnej bazy danych kontrolę poprawności danych.

Zapewniając zachowanie spójności danych przy współbieżnym dostępie, w tworzonym systemie zastosowano przetwarzanie transakcyjne oraz mechanizm blokad optymistycznych dla obiektów encji. Jego zadaniem jest zgłaszanie błędu i niedopuszczenie do zapisania nieaktualnych danych w sytuacji, gdy od czasu pobrania odpowiedniego dla encji rekordu danych z bazy danych do momentu próby jego zapisu, rekord był równolegle modyfikowany np. przez innego użytkownika. Zastosowane transakcje zapewniają przetwarzanie operacji w niepodzielnych grupach z zapewnieniem atomowości, spójności, izolacji i trwałości, cechy te określa akronim ACID (Atomicity, Consitency, Isolation, Durability). Zastosowano przy tym strategię CMT

(Containter Managed Transation), co spowodowało, że kontener EJB przejął zarządzanie granicami transakcji. Ustalono, że wraz z każdym wywołaniem API warstwy logiki biznesowej rozpoczyna się nowa, niezależna transakcja.

W implementacji przypadków użycia w tworzonym systemie zapewniono obsługę błędów. W tym celu zaimplementowane zostały własne klasy wyjątków, które rozszerzają nadrzędną klasę bazową, która reprezentuje wyjątki aplikacyjne. W implementowanych przypadkach użycia niejednokrotnie identyfikowano wiele scenariuszy błędów, co wymagało przygotowania wyjątków dla każdej sytuacji błędu. W każdej zaimplementowanej klasie wyjątku utworzono metody fabrykujące, które adekwatnie do napotkanej sytuacji błędu tworzą wyjątek danej klasy wraz z odpowiednim kluczem internacjonalizacji i ewentualnie z jego przyczyną. Większość tworzonych wyjątków przechwytywana jest na poziomie udostępnianego dla warstwy prezentacji API poprzez specjalnie utworzoną klasę. Wysłany zostaje wówczas kod odpowiedzi 400, wraz z kluczem błędu pobranym z wyjątku, który zostanie następnie przekształcony w odpowiedni komunikat, zgodnie z ustaloną wersją językową interfejsu użytkownika.

Niezbędnym elementem systemu jest dziennik zdarzeń, który utrwala historię podstawowych zdarzeń w systemie np. wywołań metod biznesowych komponentów EJB, granic transakcji wraz rezultatem ich zakończenia oraz błędów jakie wystąpiły w trakcie działania systemu. Dla każdego rejestrowanego zdarzenia zapisywana jest identyfikacja konta uwierzytelnionego użytkownika oraz znacznik bieżącego czasu Należy zauważyć, że wymagane jest zapewnienie aktualnego znacznika czasu w systemie operacyjnym serwerów obsługujących system. Zarejestrowana historia systemu jest później niezbędna w diagnostyce błędów czy też kontroli odpowiedzialności, gdzie rozliczane są działania poszczególnych użytkowników. Zastosowano dziennik zdarzeń utrzymywany przez serwer aplikacyjny.

5 Implementacja interfejsu użytkownika

Zastosowanie przeglądarki internetowej jako oprogramowania klienta niezbędnego do wyświetlenia interfejsu użytkownika wymaga zabezpieczenia komunikacji z serwerem, który udostępnia interfejs użytkownika osadzony w stronach WWW. Dzięki wykorzystaniu protokołu HTTPS i certyfikatu SSL, komunikacja jest szyfrowana zgodnie ze standardem SSL/TLS. Poprawne uwierzytelnienie serwera wymaga podpisanego przez niezależną jednostkę CA certyfikatu. Wspomniane certyfikaty są kluczowym punktem bezpiecznej komunikacji, gdyż zawierają one informacje związane z właścicielem

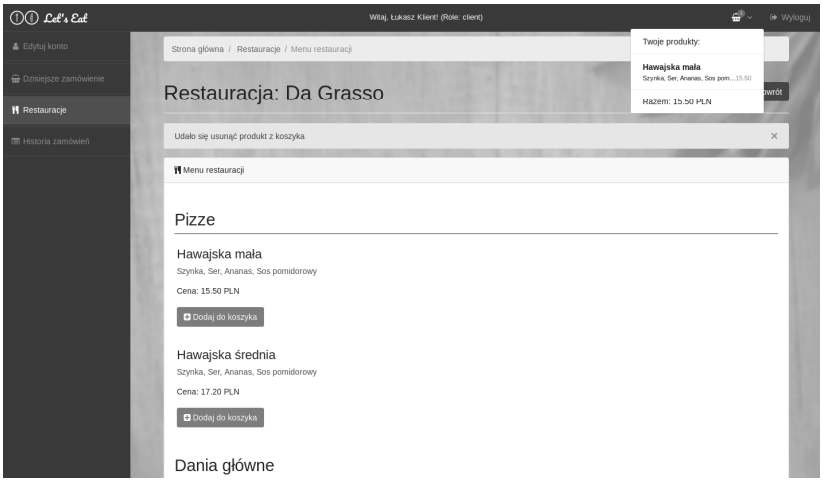
domeny, która występuje w adresie URL interfejsu użytkownika stworzonego systemu.

Do uwierzytelniania zastosowano mechanizm udostępniany przez serwer aplikacyjny Payara. Na stronie głównej aplikacji użytkownik musi podać swój adres e-mail i hasło, a następnie kliknąć przycisk zaloguj. Dane trafiają wtedy do serwera aplikacyjnego, który weryfikuje je z danymi odczytanymi z widoku bazodanowego. Widok ten udostępnia adresy e-mail użytkowników jako loginy i skróty wyliczone algorytmem SHA256 z haseł. W przypadku pomyślnego uwierzytelnienia, użytkownik zostanie powiązany z posiadanymi poziomami dostępu i przekierowany do głównego interfejsu - jeżeli natomiast dane uwierzytelniania nie zostały dopasowane do wzorca, wówczas użytkownik zostanie przekierowany na stronę ze stosownym komunikatem.

W stworzonej aplikacji internetowej interfejs użytkownika zbudowany został na podstawie szablonu strony WWW, co zapewnia ujednolicenie wyglądu stron. W szablonie strony wydzielono: nagłówek, menu boczne oraz stopkę - w pozostałym miejscu ładowany jest dynamicznie odpowiedni widok (Rys. 4).

Rys. 4. Szablon stron autorskiego systemu po uwierzytelnieniu

Autoryzacja kont w warstwie prezentacji jest wykonywana niezależnie od pozostałych warstw. Nagłówek interfejsu użytkownika zawiera przycisk służący do wylogowania oraz informacja o zalogowanym użytkowniku w postaci jego imienia i nazwiska wraz z posiadanymi poziomami dostępu. Menu boczne interfejsu użytkownika dostosowuje się do poziomów dostępu, jakie są przypisane do konta. W przypadku klientów, po wybraniu dostawcy posiłku przy składaniu zamówienia, na pasku górnym pojawia się także koszyk pokazujący elementy kompletowanego zamówienia (Rys. 5).



Rys. 5. Interfejs panelu klienta podczas dodawania posiłku do koszyka

Należy zauważyć, że w stworzonym systemie błędy zgłaszane przez warstwę logiki są przechwytywane i obsługiwane na poziomie kontrolerów AngularJS, a następnie poprzez specjalnie utworzony serwis dodawane są komunikaty w interfejsie użytkownika. Przykład takiego wyświetlonego komunikatu błędu przedstawia Rys. 5.

Interfejs użytkownika cechuje się responsywnością, co osiągnięto poprzez zastosowanie platformy programistycznej Bootstrap [5]. Zapewnia ona możliwość budowania stron złożonych z dwunastokolumnowej siatki, która dostosowuje się do ekranu urządzenia. W każdym przecięciu wiersza i kolumny w siatce rozmieszczenia można umieścić inny element interfejsu użytkownika. Rys. 6 przedstawia przykład wyświetlenia strony na mniejszym ekranie smartfona.

System "Let's Eat" został przygotowany w polskiej i angielskiej wersji językowej. Internacjonalizacja interfejsu użytkownika zrealizowana została w kodzie JavaScript. Tłumaczenia wszystkich użytych komunikatów zapisano w plikach zawierających obiekty, gdzie odpowiednim kluczom przypisuje się wartości tłumaczeń w danym języku. Następnie w poszczególnych widokach powołujemy się na klucze zawarte we wspomnianych wcześniej obiektach. Podczas renderowania strony, klucz zostanie zamieniony na tłumaczenie w obowiązującym języku ustalonym dla przeglądarki internetowej użytkownika. W systemie zapewniono możliwość wybrania przez użytkownika wersji językowej dla interfejsu użytkownika. Przy braku wyboru użytkownika podczas pierwszego wyświetlenia strony sprawdzone zostają preferencje językowe przeglądarki - jeżeli dla preferowanego języka dostępne są tłumaczenia to zostanie on

zastosowany, w przeciwnym razie użyty zostanie użyty domyślny język angielski.

Rys. 6. Wygląd interfejsu użytkownika systemu wyświetlonego na ekranie smartfona

6 Podsumowanie

Realizując projekt systemu wspomagającego grupowe zamawianie posiłków dla pracowników przedsiębiorstwa stworzono relacyjną bazę danych wraz z aplikacją internetową. Zapewniono przy tym obsługę pięciu poziomów dostępów o zróżnicowanej funkcjonalności. W systemie zastosowano architekturę trójwarstwową, co zapewniło jego skalowalność. W szczególności zapewniono separację implementacji reguł przetwarzania danych od interfejsu użytkownika w aplikacji internetowej, który jest dostarczany przez dynamicznie generowane strony WWW. Zachowano przy tym ujednolicony wygląd interfejsu użytkownika poprzez zastosowanie szablonu strony.

Zastosowane przy budowie systemu technologie umożliwiają jego uruchomienie w różnych platformach programowych. Dzięki zachowaniu standardów technologicznych w implementacji aplikacji internetowej, możliwe jest jej uruchomienie w dowolnym serwerze aplikacyjnym zapewniającym wymagane kontenery Java EE. Dodatkowo podział na warstwy i moduły ułatwia dalszy rozwój stworzonego systemu, co zapewnia elastyczność stworzonego oprogramowania. W każdej z warstw systemu zastosowano niezależne mechanizmy kontroli

poprawności danych jak i kontroli dostępu do danych. W przypadku warstwy prezentacji zastosowano walidację wraz z regułami kontrolującymi dostęp do interfejsu użytkownika zgodnie z przydzielonymi dla jego konta poziomami dostępu. W warstwie logiki biznesowej zastosowano mechanizmy kontroli dostępu do metod komponentów EJB, w których zaimplementowano zasady przetwarzania danych. Jednocześnie dla obiektów encji, reprezentujących rekordy z relacyjnej bazy danych, zastosowano dodatkowe mechanizmy weryfikacji poprawności danych. W warstwie składowania danych zapewniono poprawność danych poprzez zastosowanie ograniczeń w strukturach relacyjnej bazy danych. Funkcjonalność aplikacji została podzielona na moduły, a każdy z nich korzysta z innego konta dostępowego do relacyjnej bazy danych. Dla każdego konta bazodanowego przypisano inne uprawnienia do tabel zawartych w bazie. W utworzonym systemie zastosowano uwierzytelnianie użytkowników z wykorzystaniem haseł, przy czym wzorce haseł są przechowywane w relacyjnej bazie danych w postaci niejawnej.

Utworzony system informatyczny jest wielodostępny, dlatego w budowie systemu uwzględniono mechanizmy odpowiadające za kontrolę spójności danych. Wszystkie operacje na danych są wykonywane w granicach transakcji, przy czym uwzględniono separację danych przy równolegle przetwarzanych transakcjach. Dodatkowo zastosowano mechanizm blokad optymistycznych, który uniemożliwia nadpisanie danych w relacyjnej bazie danych przez nieaktualną ich wersję. Zastosowane mechanizmy umożliwiają równoległy dostęp i przetwarzanie danych bez naruszenia reguł poprawności danych w stworzonym systemie informatycznym.

Uwzględniając charakterystykę pracy stworzonego systemu grupowego zamawiania posiłków zapewniono zachowanie historii wykonywanych operacji w dziennikach zdarzeń. Rejestrowane podlegają wywołania metod komponentów EJB z warstwy logiki biznesowej, granice transakcji aplikacyjnych oraz błędy. Zachowana historia działania systemu umożliwia kontrolę odpowiedzialności, ponieważ dla każdego zdarzenia rejestrowana jest również tożsamość konta, z którego były wykonane działania powodujące wystąpienie zdarzenia. W każdym z zaimplementowanych przypadków użycia uwzględniono obsługę błędów, stąd oprócz zarejestrowania zdarzenia błędu w dzienniku zdarzeń system dostarcza użytkownikowi czytelnej informacji o wystąpieniu błędu. Komunikaty błędów, podobnie jak cały interfejs użytkownika zostały poddane internacjonalizacji. Oznacza to, że interfejs użytkownika stworzonego systemu obsługuje różne wersje językowe, dobór wersji językowej jest dostosowywany automatycznie do preferencji językowych przeglądarki internetowej użytkownika.

Zapewniono obsługę języka polskiego i angielskiego, z możliwością dodania innych wersji językowych.

Stworzony system wyróżnia się wśród konkurencyjnych systemów zamawiania posiłków, oferując oprócz możliwości zamawiania posiłków od różnych dostawców wspomaganie rozliczeń finansowych tak złożonych zamówień. Obsługa wymienionych funkcjonalności okazała się kluczowa dla zamawiającego stworzony system informatyczny. Należy zauważyć, że zastosowanie stworzonego systemu jest możliwe dla przedsiębiorstw o różnej specyfice - zarówno, gdy organizacja w całości lub częściowo finansuje posiłki własnym pracownikom.

Bibliografia

[1] Dybikowski Z., PostgreSQL, Helion, Gliwice, 2012
[2] Kalbarczyk D., Kalbarczyk A., AngularJS. Pierwsze kroki, Helion, Gliwice, 2015
[3] Kortas M., Bootstrap. Praktyczne projekty, Helion, Gliwice, 2016
[4] Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, and David J. Lilja. 2000. Techniques for obtaining high performance in Java programs. ACM Comput. Surv. 32, 3, 213-240.
[5] Brunnert A., Vögele C., Krcmar H. (2013) Automatic Performance Model Generation for Java Enterprise Edition (EE) Applications. In: Balsamo M.S., Knottenbelt W.J., Marin A. (eds) Computer Performance Engineering. EPEW 2013. Lecture Notes in Computer Science, vol 8168. Springer, Berlin, Heidelberg
[6] Matthias K., Kane S. P., Docker. Praktyczne zastosowania, Helion, Gliwice, 2017
[7] Yener M., Theedom A., Java EE. Zaawansowane wzorce projektowe, Helion, Gliwice, 2015
[8] Daoqi Yang, Java Persistence with JPA, Outskirts Press, 2010
[9] Keith M., Schincariol M. (2013) Introduction. In: Pro JPA 2. Apress, Berkeley, CA
[10] Thomas A.: Selecting Enterprise JavaBeans Technology, WebLogic, Inc., Boston, MA, 1998.

GROUP MEAL ORDERING SYSTEM FOR ENTERPRISE EMPLOYEES

Nowadays, the labor market is oriented on efficiency and high quality of services. The synergy of these factors is visible in various areas and affects the organization of workplaces, which is significantly noticeable in area of corporations. Optimizations and organizational improvements of the work model translate into decreasing costs and increasing competitiveness. One of the most recurring problems among employers with a large team of employees is the daily process of meal ordering from external food suppliers by employees. The inefficient information flow between employees, employers and food suppliers has an impact on reducing work efficiency. The problem is even more severe for the employer, who offers the possibility of partial or full meals financing for employees, because financial issues have to be taken into account as well. The authors presented a solution to the mentioned problem by creating an original IT system supporting the group meal ordering by the employees of the company.

Keywords: Java Enterprise Edition, AngularJS, group meal ordering