Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Paweł Wrzeszcz
Nr albumu: 209308

System do uzgadniania terminów spotkań

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr Janiny Mincer-Daszkiewicz
Instytut Informatyki

Grudzień 2007

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Streszczenie

Tematem pracy jest system automatyzujący proces wyboru optymalnego terminu spotkania, w którym może uczestniczyć wiele osób z różnymi preferencjami.

W ramach analizy przedstawiam możliwe podejścia do pojęcia *optymalnego terminu*, sposoby definiowania indywidualnych preferencji oraz algorytm wyboru optymalnego terminu. Omawiam wymagania stawiane systemowi oraz aktualnie stosowane metody i aplikacje służące do uzgadniania terminów spotkań.

Całość rozważań poparta jest analizą sposobu implementacji systemu, którego opis prezentuję w drugiej części pracy. Komunikacja użytkownika z systemem, w tym definiowanie parametrów spotkania oraz składanie preferencji, odbywa się na stronie internetowej zbudowanej z użyciem technologii AJAX, działającej na serwerze aplikacji JEE. Za wybór optymalnego terminu odpowiada biblioteka operująca na zbiorach interwałów czasowych.

Słowa kluczowe

spotkanie, uzgadnianie, termin, Java, JEE, EJB 3.0, AJAX, JBoss Seam

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

H. Information Systems
H.4 INFORMATION SYSTEMS APPLICATIONS
H.4.1 Office Automation

Tytuł pracy w języku angielskim

Event Scheduling System

Spis treści

Wprowadzenie

Ustalanie dat i godzin różnych spotkań jest zadaniem, z którym wiele osób ma do czynienia codziennie. W grupie kilku osób wybranie terminu dogodnego dla wszystkich zazwyczaj nie stanowi większego problemu. Natomiast w sytuacji, kiedy spotkanie dotyczy większej liczby uczestników, jego termin jest z reguły narzucany z góry, bądź też ustalany na podstawie konsultacji, jednak nie zawsze z uwzględnieniem preferencji wszystkich uczestników spotkania. Sytuacja ta często wynika z braku fizycznej możliwości wzięcia pod uwagę dużej liczby preferencji przez osobę wyznaczającą termin spotkania. Pożądane byłoby zatem zautomatyzowanie tego procesu.

Istnieją co prawda aplikacje pełniące funkcje kalendarzy osobistych, umożliwiające udostępnianie innym osobom informacji o tym, kiedy jest się zajętym. Część narzędzi tego typu pozwala na wyświetlenie na jednej stronie kalendarzy wielu osób, co może być pomocne przy ustalaniu terminu spotkania, jednak sprawdza się głównie dla niewielkich grup. Korzystanie z takich rozwiązań wymaga także, aby wszystkie zainteresowane spotkaniem osoby sumiennie prowadziły swoje kalendarze i używały kompatybilnych aplikacji.

Częściowe rozwiązanie tych problemów oferują serwisy internetowe pozwalające na tworzenie stron do głosowania na poszczególne terminy spotkania. Główna wada tego typu rozwiązań to brak możliwości dokładnego i elastycznego składania preferencji. Dla każdego zaproponowanego terminu można wypowiedzieć się tylko, czy jest się wtedy zajętym, czy nie. Nie istnieje możliwość zgłoszenia preferencji mówiącej np. o tym, że pasuje nam spotkanie się pół godziny później niż zaproponowany termin. Ponadto istniejące aplikacje nie oferują wizualizacji terminów spotkania z uwzględnieniem ich popularności.

W ramach niniejszej pracy zaprojektowano i wykonano system pozwalający na zautomatyzowanie procesu wyboru terminu spotkania, niejako łączący w sobie funkcjonalność kalendarza z aplikacją służącą do głosowania. Projektując serwis jako główne wymaganie przyjęto możliwość elastycznego składania preferencji. Umożliwienie uczestnikom podawania przedziałów czasowych, w których mogą przybyć na spotkanie, jest rozwiązaniem idącym o krok dalej w stosunku do koncepcji zwykłego głosowania na poszczególne terminy. Dla osiągnięcia tego założenia niezbędne było opracowanie interfejsu użytkownika służącego składaniu dokładnych preferencji. Dodatkowym wymaganiem dotyczącym interfejsu była wizualna prezentacja popularności terminów.

System umożliwia zdefiniowanie spotkania, tj. podanie możliwych terminów i czasu trwania spotkania, oraz zebranie preferencji od jego uczestników. Możliwe terminy oraz ich popularność są w przystępny sposób prezentowane na kalendarzu. Składanie preferencji odbywa się poprzez zaznaczanie na kalendarzu przedziałów czasowych, w których dana osoba może stawić się na spotkanie. Aplikacja pozwala na późniejszą zmianę preferencji oraz ustawień spotkania. Preferencje uczestników są na bieżąco analizowane celem znalezienia optymalnego terminu dla spotkania oraz zaprezentowania aktualnej popularności poszczególnych przedziałów czasowych.

Dodatkową cechą systemu jest możliwość zdefiniowania niektórych uczestników spotkania

jako takich, którzy bezwzględnie muszą uczestniczyć w spotkaniu. System bierze pod uwagę te ustawienia przy wyznaczaniu optymalnego terminu spotkania.

Serwis jest napisany w języku Java i działa na serwerze aplikacji JBoss Application Server. Za wybór optymalnego terminu odpowiedzialna jest biblioteka operująca na różnego rodzaju uporządkowanych zbiorach interwałów czasowych. Całość komunikacji z użytkownikiem jest realizowana poprzez dynamiczną stronę WWW. Warstwa prezentacji została wykonana w technologii AJAX. Do budowy aplikacji użyto szkieletu programistycznego JBoss Seam oraz technologii Enterprise JavaBeans. Kalendarz wyświetlany na stronie został zbudowany przez rozszerzenie komponentu pochodzącego z otwartej biblioteki MyFaces Tomahawk.

W rozdziale 1 zawarto opis problemu oraz zaproponowano klasyfikację aktualnie istniejących rozwiązań. Rozdział 2 to analiza wymagań stawianych systemowi oraz możliwych sposobów ich realizacji, począwszy od minimalnej funkcjonalności po dodatkowe cechy systemu. W rozdziale 3 przedstawione zostały technologie wykorzystane podczas implementacji aplikacji. Rozdział 4 zawiera projekt systemu oraz opis implementacji jego poszczególnych modułów. W rozdziale 5 opisano wygląd interfejsu użytkownika, natomiast w rozdziale 6 przedstawiono kilka przykładów zastosowań aplikacji w życiu codziennym. W ramach podsumowania pracy (rozdział 7) przedyskutowano także możliwe rozszerzenia systemu.

Rozdział 1

Systemy wspomagające uzgadnianie terminów spotkań

1.1. Optymalny termin

1.1.1. Problem wyboru terminu

Zadanie jakie przed sobą stawiamy polega na tym, aby organizując spotkanie dobrać termin odpowiedni dla jak największej liczby jego uczestników, a najlepiej odpowiadający wszystkim zainteresowanym osobom. Kwestie, które będą nas interesowały to definicja samego spotkania oraz jego uczestnicy i ich preferencje czasowe.

Definicja spotkania to przede wszystkim *lista terminów*, w których to spotkanie może się odbyć. Ponadto musimy znać *czas trwania* spotkania. Spotkanie definiuje zwykle jego *organizator*.

Dla każdego uczestnika spotkania będą nas interesowały jego *preferencje czasowe*, czyli lista przedziałów czasowych, w których dana osoba może pojawić się na spotkaniu.

1.1.2. Definicja optymalnego terminu

Załóżmy, że mamy zdefiniowane spotkanie, tj. znamy terminy, w których może się ono odbyć, i czas trwania oraz, że mamy dane preferencje czasowe dla każdego uczestnika. Przez *optymalny termin* będziemy rozumieli przedział czasu, równy długości spotkania, w którym wszyscy uczestnicy spotkania mogą się na nie stawić. W przypadku gdy taki termin nie istnieje, za optymalny będziemy uważali termin, w którym na spotkaniu może się znaleźć maksymalna liczba osób.

1.2. Sposoby wyboru terminu

W życiu codziennym ludzie dokonują wyboru terminów spotkań na różne sposoby. Może być to wybór terminu optymalnego lub bliskiego optymalnemu, także według innej miary niż podana powyżej. W dalszej części tego punktu prezentuję krótki przegląd sposobów wyboru terminu z podziałem na metody manualne oraz korzystające z oprogramowania komputerowego, które może ten proces ułatwić.

1.2.1. Wybór samodzielny

Przez wybór samodzielny rozumiem ustalenie terminu spotkania bez pomocy przeznaczonych do tego celu programów komputerowych czy aplikacji internetowych. Może to oznaczać po prostu narzucenie terminu z góry lub też wybranie go po mniej lub bardziej dokładnych konsultacjach z osobami zainteresowanymi.

Narzucenie terminu z góry

Najłatwiejszym sposobem wybrania terminu jest narzucenie daty spotkania z góry. Metoda ta ma zastosowanie szczególnie w przypadku dużej liczby uczestników spotkania lub gdy termin ustala osoba stojąca wyżej w hierarchii niż reszta uczestników spotkania.

Wymaga to oczywiście dostosowania się pozostałych osób do decyzji. Takie postępowanie może prowadzić do tego, że część osób nie będzie mogła wziąć udziału w spotkaniu. Może też się okazać, że istniał termin, który pasował o wiele większej liczbie osób, niż ten, który został wybrany arbitralnie.

Publiczne konsultacje

Innym rozwiązaniem jest wspólne przedyskutowanie możliwych terminów spotkania z częścią lub wszystkimi jego uczestnikami. Można tego dokonać osobiście, bądź za pośrednictwem środków komunikacji elektronicznej. Dobrym przykładem jest tutaj zapytanie o preferencje wysłane na forum internetowe, na którym każdy zainteresowany może wypowiedzieć się publicznie. Miejscem takiej dyskusji równie dobrze może być grupa dyskusyjna czy IRC. Można też po prostu wysłać list pocztą elektroniczną do wszystkich uczestników.

Wadą przytoczonego rozwiązania jest to, że duża grupa osób może mieć problem z osiągnięciem konsensusu w dyskusji, szczególnie w przypadku dużej rozbieżności preferencji pomiędzy uczestnikami spotkania. Ponadto taka komunikacja często jest asynchroniczna w tym sensie, że, przykładowo, niektórzy uczestnicy mogą przeczytać pytanie o preferencje szybciej niż inni. Może to spowodować sytuację „kto pierwszy, ten lepszy". Osoby, które szybko wypowiedzą się na forum i zgłoszą swoje preferencje właściwie wskażą już datę spotkania, a preferencje reszty osób nie będą wzięte pod uwagę.

Ta forma uzgadniania terminu ma zatem zastosowanie dla spotkań niewielkiej liczby osób. Wskazane jest też, żeby wszystkie zainteresowane osoby mogły uczestniczyć w konsultacjach dotyczących terminu spotkania w tym samych czasie.

Zebranie preferencji

Oczywiście zamiast prowadzenia publicznej dyskusji, zawsze można ręcznie zebrać i dokładnie przeliczyć preferencje wszystkich uczestników spotkania celem znalezienia optymalnego terminu. Między innymi dlatego, iż jest to czynność żmudna i czasochłonna, powstał system opisany w tej pracy.

1.2.2. Istniejące aplikacje

Istniejące programy komputerowe oraz serwisy internetowe wspomagające proces wyboru optymalnego terminu można podzielić na dwie grupy. Pierwsze to aplikacje działające jako kalendarze, z możliwością informowania innych osób o tym, kiedy jest się zajętym. Druga grupa rozwiązań to serwisy internetowe umożliwiające składanie preferencji poprzez proste głosowanie na podane terminy.

Kalendarze

Niektóre aplikacje pełniące rolę kalendarza osobistego udostępniają, za zgodą użytkownika, informację o tym, kiedy osoba z nich korzystająca jest zajęta. Często można po prostu dodać sobie udostępnione kalendarze innych osób do swojego własnego terminarza. Otrzymujemy w ten sposób podgląd na to, kiedy pozostałe osoby mają wolny czas i łatwiej jest wybrać termin spotkania.

Takie rozwiązanie jest bardzo wygodne dla małej grupy osób, pod warunkiem, że wszystkie te osoby korzystają z takich samych lub kompatybilnych aplikacji kalendarza. W przeciwnym wypadku nie ma jak uwzględnić preferencji osób, które używają niezgodnych ze sobą aplikacji. Pominięte zostaną także preferencje tych osób, które w ogóle nie korzystają z takiego oprogramowania, bo na przykład planują swój czas korzystając z kalendarza książkowego albo w ogóle nie korzystają z kalendarza.

Podsumowując, takie podejście jest idealnym rozwiązaniem dla małego zespołu pracowników, używających kompatybilnych ze sobą aplikacji. Trudno jest natomiast wdrożyć takie rozwiązanie mając do czynienia z większą liczbą osób.

W dalszej części punktu prezentuję krótką listę dostępnych na rynku programów pełniących rolę kalendarza, wraz z podziałem na aplikacje stacjonarne oraz internetowe. Przegląd ten nie ma na celu pełnego opisu wszystkich aplikacji, lecz wskazanie ciekawych opcji tych programów, ułatwiających uzgadnianie terminów spotkań.

Kalendarze stacjonarne

Do najpopularniejszych programów komputerowych oferującym swoim użytkownikom prowadzenie kalendarza należy zaliczyć Microsoft Outlook, Lotus Notes, Mozilla Sunbird oraz aplikację iCal. Najnowszym programem tego typu jest Windows Calendar.

Microsoft Outlook [31] jest programem funkcjonującym jako klient poczty elektronicznej oraz organizator informacji osobistych, w tym jako kalendarz. Posiada on możliwość dodawania udostępnionych kalendarzy do swojego terminarza oraz wsparcie przy planowaniu spotkań. Do skorzystania z tych funkcji potrzebna jest jednak dodatkowa usługa, Microsoft Exchange Server.

Lotus Notes [28] to produkt firmy IBM, podobnie jak Microsoft Outlook łączący w sobie funkcję klienta poczty elektronicznej z organizerem. Program ten posiada ciekawą opcję wizualnego wyboru terminu spotkania na podstawie tabeli z informacją kiedy wybrane z listy osoby są zajęte.

Mozilla Sunbird [35] jest darmową, wieloplatformową aplikacją o otwartym kodzie źródłowym. Posiada możliwość importowania i eksportowania kalendarzy w formatach iCal oraz CalDAV (patrz „Protokoły"). Współpraca z Google Calendar jest możliwa po zainstalowaniu odpowiedniej wtyczki.

Użytkownicy systemu operacyjnego Mac OS X mogą korzystać z programu iCal [18]. Aplikacja ta pozwala na dodawanie udostępnionych publicznie kalendarzy do widoku terminarza oraz na eksport własnego kalendarza poprzez protokół WebDAV. Program iCal jest tez implementacją standardu iCalendar (patrz „Protokoły").

Windows Calendar [43], będący składnikiem systemu operacyjnego Windows Vista, posiada funkcjonalność i wygląd zbliżony do programu iCal. Daje możliwość publikowania i subskrybowania kalendarzy, natomiast nie zapewnia migracji danych z innych programów [40].

Kalendarze internetowe

Najbardziej znanym kalendarzem internetowym jest Google Calendar. Jego konkurencję stanowią m.in. kalendarz amerykańskiego serwisu internetowego Yahoo! oraz witryna 30boxes.com. Wszystkie wymienione serwisy są darmowe. Pozwalają one nie tylko na tworzenie własnego kalendarza, ale także na jego udostępnianie i na oglądanie upublicznionych kalendarzy.

Google Calendar [14], oprócz swojej podstawowej funkcjonalności terminarza, zapewnia eksport zawartości w formatach RSS oraz iCalendar (patrz „Protokoły"). W Google Calendar jest możliwe wysyłanie zaproszeń na spotkania, ale bez możliwości pytania uczestników o ich preferencje czasowe – data jest ustalana z góry. Kalendarz oferowany przez firmę Google jest produktem zbudowanym w pełni w technologii AJAX (patrz rozdział 3.2).

Yahoo! Calendar [44] jest nieco uboższą aplikacją. Na dzień pisania tej pracy, kalendarz firmy Yahoo! umożliwiał oglądanie publicznych kalendarzy, ale bez agregacji wielu kalendarzy na jednej stronie. Brak jest też możliwości eksportu kalendarza do któregoś ze standardowych formatów (patrz „Protokoły"). Yahoo! Calendar jest wykonany jako dynamiczna strona WWW, w której każda akcja użytkownika powoduje przeładowanie strony.

Witryna 30boxes.com [1] dostarcza swoim użytkownikom organizator zorientowany na współpracę grupową. Częścią tej aplikacji jest kalendarz, który jako jedyny spośród wymienionych, umożliwia zarządzanie dostępem do treści za pomocą mechanizmu znaczników. Jest możliwa agregacjia wielu kalendarzy na jednej stronie oraz eksport zawartości kalendarza w formatach RSS i iCal. Przejrzysty i przyjazny interfejs użytkownika wykorzystuje technologię AJAX.

Protokoły

Do protokołów pozwalających na udostępnianie kalendarzy w internecie należy zaliczyć protokoły iCalendar, CalDAV oraz Web Calendar Access Protocol. Ponadto warto wspomnieć o reprezentacji zawartości kalendarza w formie RSS, używanej przez Google Calendar.

Standard iCalendar [10] definiuje tekstowy protokół pozwalający użytkownikom wymieniać między sobą zawartości kalendarzy. Protokół ten pozwala także na wysyłanie i otrzymywanie zaproszeń. Poszczególne elementy tego protokołu pozwalają na przesłanie informacji m.in. o konkretnym wydarzeniu, zadaniu do wykonania oraz o tym w jakich momentach użytkownik kalendarza jest zajęty.

Protokół CalDAV [9] bazuje na standardzie WebDAV, będącym rozszerzeniem protokołu HTTP. Rywalizuje on z protokołem Web Calendar Access Protocol [42], opartym wyłącznie na poleceniach HTTP GET. Odpowiedzi w tym protokole mogą mieć postać zwykłego tekstu, bądź pliku XML.

Firma Google opracowała swój własny format danych zawartych w kalendarzu Google Calendar, oparty na specyfikacji RSS. Informacje z kalendarza zawarte są w pliku XML, który może zostać odczytany przez dowolny czytnik kanałów RSS.

Aplikacje zbierające preferencje

Wymienione aplikacje przeznaczone były głównie do zarządzania prywatnymi informacjami, a możliwość agregacji kalendarzy, ułatwiająca wybranie terminu spotkania, była ich dodatkową cechą. Innym podejściem jest zebranie od wszystkich uczestników spotkania preferencji z użyciem przeznaczonego do tego celu narzędzia.

Opisaną funkcjonalność oferuje wiele stron internetowych. Należą do nich m.in. serwis Doodle [12], Meeting Wizard [30] oraz Meet-O-Matic [29]. Serwisy te udostępniają proste

narzędzia do uzgadniania terminu spotkania poprzez głosowanie na podane terminy. Definiując spotkanie, użytkownik podaje listę dat, na które później głosują uczestnicy spotkania. Głosowanie odbywa się na stronie spotkania, do której uczestnicy uzyskują dostęp dzięki specjalnemu odsyłaczowi z unikatowym identyfikatorem spotkania. System wyświetla listę osób, które dotychczas oddały głos oraz sumę głosów oddanych na poszczególne terminy. Dodatkowo, podczas wyboru najlepszej daty, aplikacja Meet-O-Matic daje możliwość oznaczenia niektórych osób jako takich, które bezwarunkowo muszą uczestniczyć w spotkaniu.

Głosowanie na podane terminy jest rozwiązaniem łatwym w użyciu i pozwalającym ustalić termin spotkania większej grupie osób bez konieczności synchronizowania danych z prywatnych kalendarzy wielu osób. System ten nie wymaga od uczestników spotkania korzystania z konkretnej aplikacji, w której wszyscy prowadziliby rejestr swoich spotkań. Wadą tego typu głosowania jest to, że trudno jest zbierać dokładne preferencje jeśli wybór terminu pozostawia pewną swobodę czasową. Załóżmy, że chcemy wybrać godzinny przedział czasowy spośród kilku dłuższych interwałów. Wśród nich znajduje się przedział długości 3 godzin, powiedzmy między godziną 12.00 a 15.00. Ustalmy, że wybieramy datę z dokładnością do 15 minut. Chcąc zebrać dokładne preferencje, musielibyśmy podać jako możliwe terminy wszystkie daty od godziny 12.00 do 15.00, co 15 minut. Nie byłoby to zbyt wygodne rozwiązanie.

Tego typu wymóg ma niebagatelne znaczenie, ponieważ jeśli ograniczymy się do podania godziny 12.00 jako opcji, osoby które mogłyby przybyć na spotkanie pół godziny później nie będą miały jak zadeklarować swojej preferencji. Może się okazać, że właśnie przesunięcie spotkania o pół godziny spowoduje, że wszyscy uczestnicy będą w stanie dotrzeć na nie na czas. Brakuje zatem możliwości dokładnego definiowania swoich preferencji.

Ponadto wspomniane aplikacje nie prezentują terminów ani wyników na kalendarzu, ale jako listę opcji. Skoro wybranie terminu polega na zwykłym głosowaniu, nie ma takiej potrzeby. Jednakże kalendarz, w którym kolory odzwierciedlałyby zainteresowanie poszczególnymi terminami, pomógłby niejednemu użytkownikowi zdecydować, które terminy pasują mu najlepiej. Taka cecha byłaby szczególnie przydatna w połączeniu ze wspomnianą możliwością składania dokładnych preferencji.

Należy także wziąć pod uwagę fakt, iż spora grupa użytkowników korzysta z elektronicznych kalendarzy omówionych wcześniej w tym rozdziale. Tworząc system zbierający preferencje, warto byłoby zaproponować takim osobom możliwość szybkiego importu swoich preferencji z ich własnych kalendarzy, zamiast zmuszać ich do ponownego definiowania terminów, w których są zajęci.

Część wspomnianych usprawnień jest przedmiotem zaimplementowanej przeze mnie aplikacji. Jej dokładne wymagania sprecyzuję w następnym rozdziale.

Rozdział 2

Wymagania

Najprostszy schemat działania w trakcie uzgadniania terminu spotkania przewiduje następujące czynności:

- zdefiniowanie spotkania,
- przesłanie informacji o spotkaniu do wszystkich uczestników,
- złożenie preferencji przez uczestników,
- wybór ostatecznego terminu spotkania.

Dodatkowe akcje użytkowników związane są ze zmianą złożonych wcześniej preferencji bądź też ze zmianą parametrów spotkania. Omówię kolejno sposoby realizacji wymienionych czynności wraz z możliwymi rozwinięciami.

2.1. Definiowanie spotkania

Spotkanie definiuje jego organizator. Definiując spotkanie, organizator podaje możliwe terminy oraz czas trwania spotkania. Oprócz tego należy się spodziewać, że przydatne może być podanie informacji nie związanych bezpośrednio z terminem spotkania, takich jak jego nazwa i ewentualny opis. Wygodną opcją byłaby możliwość podania miejsca spotkania, najlepiej poprzez zaznaczenie go na mapie na stronie internetowej.

Aby maksymalnie uprościć korzystanie z serwisu, stworzenie nowego spotkania nie powinno wymagać od użytkownika logowania się do systemu.

2.1.1. Definiowanie terminów

Uczestnicy spotkania będą mogli podać swoje preferencje tylko w ramach przedziałów czasowych podanych w definicji spotkania. Ograniczenia takie mogą występować np. z powodu dostępności sali przeznaczonej na spotkanie. Powinno być możliwe podawanie terminów z różną dokładnością, np. podania tylko dni lub podania dokładnych godzin.

Z kolei w niektórych przypadkach ograniczenia dotyczące terminu spotkania mogą nie istnieć i wybór daty będzie w całości pozostawiony osobom uczestniczącym w spotkaniu. W takim przypadku należy umożliwić podanie dużego przedziału czasowego, np. tygodnia, jako tego, w którym spotkanie może się odbyć.

Funkcja definiowania listy możliwych terminów powinna być zaimplementowana w sposób zapewniający jak najprostsze korzystanie z niej. Dobrym rozwiązaniem będzie tutaj kalendarz na stronie internetowej, w którym można zaznaczyć wybrane terminy.

Dodatkową opcją mogłoby być definiowanie spotkań cyklicznych, np. cotygodniowych. Cały proces odbywałby się tak samo, z tą różnicą, że nie byłyby uwzględniane dokładne daty, a tylko dni tygodnia.

2.2. Przesłanie uczestnikom informacji o spotkaniu

Po utworzeniu spotkania system automatycznie wygeneruje specjalny odnośnik umożliwiający składanie preferencji przez uczestników spotkania oraz odnośnik umożliwiający administrowanie spotkaniem. Będą one zawierały unikatowy identyfikator spotkania. Zostaną przesłane na adres e-mail osoby tworzącej spotkanie oraz podane na stronie potwierdzającej fakt utworzenia nowego spotkania.

Organizator podaje odnośnik do spotkania wszystkim jego uczestnikom. Jest on taki sam dla wszystkich uczestników (chyba, że istnieją wśród nich osoby wyróżnione, patrz niżej). Może to zrobić przesyłając dalej e-mail, który został mu przesłany przez system, lub podać do wiadomości w jakikolwiek inny sposób.

Odnośnik administracyjny może posłużyć do zmiany ustawień spotkania. Funkcja ta zostanie omówiona dokładniej w punkcie 2.6.

Przydatnym rozszerzeniem systemu byłaby możliwość wskazania niektórych uczestników spotkania jako tzw. *uczestników krytycznych*, czyli takich którzy muszą wziąć udział w spotkaniu (np. prelegent). Przy dobieraniu optymalnego terminu pod uwagę zostaną wzięte tylko terminy, w których tacy uczestnicy mogą się stawić. Uczestników krytycznych można rozróżnić od pozostałych osób generując dodatkowy odnośnik, przeznaczony tylko i wyłącznie dla nich.

2.3. Składanie preferencji

Aby złożyć preferencje, uczestnik spotkania wchodzi na stronę spotkania używając odnośnika, który otrzymał od organizatora spotkania. Wygodnym dla użytkownika rozwiązaniem byłoby, gdyby mógł zgłosić swoje preferencje bez potrzeby logowania się w serwisie. Z drugiej strony użytkownik powinien mieć możliwość późniejszej zmiany złożonych przez siebie preferencji. Dlatego po wejściu na stronę spotkania, użytkownik zostanie poproszony o podanie imienia oraz swojego adresu e-mail. Na ten adres zostanie mu przesłany jego prywatny odnośnik, dzięki któremu będzie mógł w miarę potrzeby zmienić swoje preferencje. Odnośnik ten zostanie mu też podany po zakończeniu składania preferencji.

Składanie preferencji odbywa się poprzez zaznaczanie terminów w kalendarzu na stronie internetowej. Powinny być na nim uwidocznione terminy, które można wybrać, czyli te, które podał tworzący spotkanie. Wizualna prezentacja popularności terminów, czyli na przykład wyróżnienie kolorami terminów pasujących dużej liczbie osób spośród tych, które dotychczas złożyły preferencje, zwiększy prawdopodobieństwo, że użytkownik właśnie składający preferencje zdecyduje się na te terminy, a w konsekwencji łatwiej będzie znaleźć konsensus. Wyznaczanie najbardziej popularnych terminów odbywać się będzie poprzez analizę preferencji zgodnie z algorytmem opisanym w punkcie 2.5.

Aby współpraca z aplikacją była dla użytkownika intuicyjna, system powinien prezentować najbardziej popularne terminy na bieżąco i aktualizować je natychmiast po tym, jak uczestnik doda lub usunie jakąś swoją preferencję. Pożądana byłaby synchronizacja widoku preferencji między różnymi użytkownikami, bez potrzeby odświeżania strony spotkania. Umożliwiłoby to np. szybkie umówienie się w kilka osób na spotkanie podczas rozmowy na czacie. Wszyscy uczestnicy widzieliby prawie natychmiast preferencje składane przez pozostałe osoby.

Dodatkową opcją jest możliwość dodania komentarza na stronie spotkania przy składaniu preferencji.

2.4. Wybór terminu spotkania

Ostatecznego wyboru terminu dokonuje osoba, która utworzyła spotkanie, w wybranym przez siebie momencie. Wykorzystując odnośnik administracyjny, organizator wchodzi na stronę pozwalającą na administrację spotkaniem (patrz p. 2.6) i wybiera opcję zakończenia składania preferencji. System dokonuje analizy podanych preferencji (patrz p. 2.5) i prezentuje wyniki, wskazując najbardziej popularne terminy, z których organizator wybiera ostateczną datę.

Po wybraniu ostatecznego terminu spotkania, dalsze głosowanie nie jest możliwe. Uczestnicy spotkania, którzy wejdą na stronę spotkania po tym, jak jego data została ustalona, mogą przeglądać wszystkie dane dotyczące spotkania oraz złożone przez siebie preferencje, ale nie mogą już ich zmieniać.

Pożądanym rozszerzeniem systemu byłaby możliwość automatycznego wysłania e-maila z informacją o ustalonej dacie do wszystkich uczestników spotkania.

2.5. Metody wyboru optymalnego terminu

W podstawowej wersji znalezienie optymalnych terminów ogranicza się do obliczenia, ile osób może przybyć na spotkanie w każdym z rozważanych odcinków czasu i znalezieniu spójnego fragmentu o długości równej czasowi trwania spotkania, w którym może w nim uczestniczyć jak największa liczba osób. Dodatkowe ograniczenia, takie jak np. występowanie uczestników krytycznych, możemy obsłużyć modyfikując w niewielkim stopniu powyższy algorytm, co omawiam w dalszej kolejności.

2.5.1. Uczestnicy krytyczni

Uczestnicy krytyczni to osoby, które bezwarunkowo muszą uczestniczyć w spotkaniu. Może być to na przykład osoba prezentująca coś w trakcie spotkania.

W celu znalezienia optymalnego terminu spotkania z uwzględnieniem uczestników krytycznych, najpierw znajdujemy terminy, w których wszyscy uczestnicy, którzy bezwarunkowo muszą pojawić się na spotkaniu, mają taką możliwość. W dalszej części poszukiwania optymalnych terminów bierzemy pod uwagę tylko te terminy.

Może się jednak zdarzyć, że dodanie wymagania o uczestnikach krytycznych powoduje, że nie da się znaleźć optymalnego terminu. Aby w rankingu najbardziej popularnych terminów uwzględnić takie sytuacje, wystarczy w algorytmie wyboru optymalnego terminu traktować wszystkie możliwe terminy, w których może uczestniczyć choć jeden uczestnik krytyczny, jako ważniejsze niż te, w których żaden uczestnik krytyczny nie ma czasu.

Ponadto, organizator spotkania powinien mieć możliwość zmiany statusu danego uczestnika z krytycznego na zwykłego i odwrotnie.

2.5.2. Brak konsensusu

W sytuacji, w której nie istnieje termin odpowiadający wszystkim uczestnikom spotkania, pewną opcją jest poproszenie osób, które nie mogłyby zjawić się na spotkaniu o zrewidowanie swoich preferencji. Każdy uczestnik spotkania ma podgląd na listę uczestników wraz z informacją, które osoby mogą zjawić się na spotkaniu w konkretnych terminach.

Korzystając ze strony administracyjnej, inicjator spotkania może też w każdej chwili zmienić parametry spotkania. W celu znalezienia terminu odpowiadającego wszystkim uczestnikom, może na przykład skrócić długość spotkania w stosunku do początkowo planowanej. System wyszuka wtedy optymalne terminy na nowo. Organizator może też dodać terminy do listy terminów, w których może się odbyć spotkanie.

2.6. Administracja

W ramach zarządzania spotkaniem, jego organizator ma możliwość:

- Zmiany takich parametrów spotkania jak nazwa, opis i miejsce spotkania oraz jego długość,
- Podglądu listy uczestników, z możliwością usunięcia wybranej osoby z listy,
- Zmiany statusu danego uczestnika z krytycznego na zwykłego i odwrotnie,
- Modyfikacji listy terminów, w których może odbyć się spotkanie,
- Zakończenia składania preferencji (patrz p. 2.4),
- Usunięcia spotkania.

Usunięcie danej osoby z listy uczestników skutkuje usunięciem wszystkich złożonych przez taką osobę preferencji. Zmiana statusu osoby powoduje, że złożone przez tą osobę preferencje zaczynają być traktowane według nowego statusu. Te operacje oraz modyfikacja długości lub listy terminów spotkania mogą spowodować zmiany w liście najpopularniejszych terminów.

2.7. Integracja z innymi aplikacjami

2.7.1. Import preferencji

Opcjonalną cechą, mogącą ułatwić niektórym użytkownikom posługującym się aplikacjami udostępniającymi funkcję kalendarza (patrz p. 1.2.2) korzystanie z systemu, byłaby możliwość importu informacji o zajętych terminach bezpośrednio z ich aplikacji. W ten sposób, podczas składania preferencji, takie osoby nie musiałyby przepisywać danych ze swoich kalendarzy do systemu. Oczywiście dalsza edycja preferencji byłaby nadal możliwa.

Importu preferencji można by dokonywać korzystając ze standardowych protokołów udostępnianych przez większość aplikacji kalendarzy.

2.7.2. Zaproszenie na spotkanie

Po dokonaniu wyboru ostatecznego terminu spotkania, system mógłby wysyłać informację z wybraną datą. Pożądane byłoby ustandaryzowanie takiej informacji, tj. wysłanie jej w formacie zgodnym ze standardem iCalendar (patrz p. 1.2.2).

2.8. Interfejs użytkownika

Korzystanie z systemu powinno być łatwe i wygodne. Wysoce pożądane jest, aby praca z systemem nie powodowała przeładowania strony po każdej akcji użytkownika. Użytkownik nie

powinien być też zmuszany do otwierania coraz to nowych ekranów celem wykonania konkretnej akcji. Udostępnienie dodatkowych opcji nie powinno się odbywać kosztem przejrzystości interfejsu.

Jak wspomniano wcześniej, idealnym rozwiązaniem do prezentacji sumarycznych preferencji jest kalendarz, na którym kolory reprezentują popularność terminów. Składanie preferencji powinno być możliwe zarówno poprzez zaznaczanie przedziałów czasowych na kalendarzu, jak i przez ręczne wpisanie dat w odpowiednich polach formularza.

Możliwość składania preferencji bez konieczności logowania się do serwisu zwiększy potencjalny krąg odbiorców i ułatwi skorzystanie z systemu. Ponadto umożliwi to szybkie zaznajomienie się z aplikacją osobom korzystającym z niej po raz pierwszy.

Architektura systemu powinna też przewidywać możliwość wyświetlania interfejsu użytkownika w wielu językach.

2.9. Wymagania niefunkcjonalne

2.9.1. Niezależność od używanego oprogramowania

Dostęp do serwisu nie powinien być uzależniony od systemu operacyjnego i rodzaju komputera, na którym pracuje użytkownik. Aby móc korzystać z serwisu, użytkownik będzie potrzebował jedynie przeglądarki internetowej Mozilla Firefox z włączoną obsługą języka JavaScript. Współpraca z innymi przeglądarkami WWW jest pożądaną, acz opcjonalną cechą systemu.

2.9.2. Wydajność i transakcyjność

System powinien reagować na działania użytkownika z możliwie najmniejszymi opóźnieniami. Obsługa akcji podejmowanych przez użytkownika nie powinna powodować przeładowywania strony serwisu. Możliwie najkrótszy powinien być także czas otwierania się strony serwisu.

Właściwości te powinny być zachowane także w przypadku korzystania z serwisu przez wielu użytkowników w tym samym czasie. Warstwa odpowiedzialna za dostęp i modyfikację danych oraz wybór optymalnego terminu także powinna być w stanie obsłużyć wiele zapytań równocześnie.

System powinien być transakcyjny. Powinno być możliwe uruchomienie aplikacji na klastrze.

2.9.3. Niezależność od bazy danych

Podsystem odpowiedzialny za dostęp do danych nie powinien zakładać konkretnej implementacji bazy danych. Upraszcza to testowanie oraz ułatwia ewentualną zmianę dostawcy bazy danych.

Rozdział 3

Użyte technologie

W tym rozdziale zostały omówione technologie zastosowane do implementacji systemu.

3.1. Java Enterprise Edition

Aplikacja została napisana w języku Java, z użyciem platformy Java Enterprise Edition 5.0 (Java EE) [20]. W skład tego standardu wchodzi ponad 20 technologii, pozwalających na tworzenie wielowarstwowych aplikacji internetowych. Użyte przeze mnie technologie, będące częścią platformy Java EE, to:

- JavaServer Faces 1.2,
- JavaServer Pages Standard Tag Library,
- Java Servlet,
- Enterprise JavaBeans 3.0,
- JavaPersistence API,
- JavaMail.

W przypadku większości specyfikacji technologii Java EE, istnieje więcej niż jedna implementacja zgodna z daną specyfikacją. Wybór konkretnej implementacji najczęściej podyktowany jest ceną (tylko część rozwiązań to produkty darmowe), wydajnością, wsparciem oferowanym przez firmę lub społeczność programistów skupionych wokół danego produktu oraz ewentualną dodatkową funkcjonalnością oferowaną przez implementację.

W swojej pracy, jako implementacji technologii Java EE wykorzystuję otwarte oprogramowanie dostarczane przez firmy JBoss i Sun Microsystems. Używam także komponentów rozwijanych pod egidą Apache Software Foundation.

3.1.1. Serwer aplikacji

Do budowy systemu użyłem serwera aplikacji JBoss Application Server (JBoss AS) [22] w wersji 4.2.1. Jest on jednym z najszerzej używanych serwerów spośród rozwiązań darmowych [32]. JBoss AS pozwala programiście korzystać z technologii Enterprise JavaBeans 3.0 (EJB3).

Technologia ta definiuje sposób implementacji „standardowych komponentów po stronie serwera dla rozproszonych aplikacji biznesowych”[1] [7]. Budowanie aplikacji z komponentów

[1] Tłumaczenie własne autora.

EJB3 daje nam skalowalność, transakcyjność oraz realizuje równoczesny dostęp do komponentów przez wielu użytkowników [11]. Komponenty EJB3 można definiować przy użyciu anotacji[2] lub w pliku w formacie XML.

Częścią standardu EJB3 jest JavaPersistence API. Interfejs ten pozwala na dostęp do bazy danych w sposób niezależny od konkretnego jej dostawcy. JPA definiuje odwzorowanie między obiektami Javy a tabelami w relacyjnej bazie danych. Pozwala to programiście tworzyć aplikacje właściwie bez potrzeby bezpośredniego odwoływania się do bazy danych. Implementacja JavaPersistence API w JBoss AS bazuje na technologii Hibernate [16].

Jako kontener serwletów JBoss AS wykorzystuje, nieco zmodyfikowaną, wersję serwera Apache Tomcat [3].

3.1.2. JavaServer Faces

Technologia JavaServer Faces (JSF) [8] służy do budowania interfejsu użytkownika z gotowych komponentów. Komponenty te komunikują się z logiką systemu poprzez zwykłe obiekty Javy (POJO). Implementacja JavaServer Faces, załączona do użytej przeze mnie wersji JBoss AS, to implementacja referencyjna tej technologii firmy Sun Microsystems.

JSF jest najczęściej używany jako nakładka na nieco starszą technologię tego typu, JavaServer Pages (JSP). JSF może być też używany samodzielnie [5] lub w połączeniu z technologią Facelets [21], oferującą daleko idące możliwości budowania stron internetowych z użyciem szablonów. Pominięcie silnika JSP przy korzystaniu z JSF, powoduje wzrost wydajności o 30-50% [45]. Autor zastosował w swojej aplikacji rozwiązanie będące połączeniem Facelets i JSF.

Istnieje wiele komponentów JSF rozszerzających standardowe kontrolki, w tym komponenty wykorzystujące technologię AJAX (patrz podrozdział 3.2). Do budowy mojej aplikacji użyłem bibliotek komponentów MyFaces Tomahawk oraz Richfaces.

3.1.3. JBoss Seam

JBoss Seam [24] nazywany jest „brakującym szkieletem" Javy EE. Seam nie jest nową technologią samą w sobie, natomiast scala takie rozwiązania jak JSF i EJB3 w jedną całość, ułatwiając programiście tworzenie aplikacji z ich użyciem. Seam rozszerza możliwości JSF w budowania aplikacji według wzorca projektowego Model-Widok-Kontroler oraz umożliwia wykorzystanie komponentów EJB3 jako obiektów integrujących widok z warstwą logiki.

JBoss Seam wprowadza pojęcie *konwersacji* z użytkownikiem, eliminując wymóg stosowania sesji do zapamiętywania podręcznych danych dotyczących użytkownika aplikacji internetowej. Korzystanie z konwersacji zwiększa wydajność systemu, co jest szczególnie ważne w przypadku stosowania technologii AJAX (patrz podrozdział 3.2). Ułatwia też klastrowanie aplikacji oraz likwiduje trudności w oprogramowaniu sytuacji, w których jeden lub wielu użytkowników ma otwartych kilka okien tej samej aplikacji w jednej przeglądarce internetowej [45].

JBoss Seam nie jest częścią standardu Java EE w wersji 5.0, natomiast wiele rozwiązań w nim zawartych znalazło swoje miejsce we właśnie opracowywanej specyfikacji „WebBeans" [41], mającej wejść w skład następnej wersji standardu Java.

W swojej aplikacji użyłem szkieletu Seam w wersji 2.0 CR2.

[2]Anotacje w języku Java pozwalają dołączać metadane do klas, pól i metod obiektów.

3.2. AJAX

AJAX, czyli Asynchroniczny JavaScript i XML, jest to technologia[3] tworzenia stron internetowych, dzięki której akcje użytkownika nie powodują załadowania całej strony od nowa. Jak sama nazwa wskazuje, korzysta ona z asynchronicznej komunikacji pomiędzy przeglądarką internetową a serwerem [45].

Rozwiązanie to upowszechniła firma Google, stosując je w swoich aplikacjach, takich jak Google Maps, Google Suggest czy Gmail. Nazwy AJAX jako pierwszy użył Jesse James Garret z firmy Adaptive Path w 2005 roku [13].

3.2.1. AJAX w JSF

Istnieje kilka sposobów korzystania z technologii AJAX w połączeniu z JSF. Można:

- Stosować gotowe komponenty już działające w technologii AJAX. Dobrym przykładem zbioru takich komponentów jest biblioteka ICEFaces [19].
- Dodać możliwość komunikacji asynchronicznej do istniejących komponentów. Pozwala na to biblioteka Ajax4JSF, będąca częścią projektu RichFaces [23].
- Ręcznie oprogramować komunikację z serwerem na poziomie języka JavaScript i serwletów.
- Skorzystać z gotowych rozwiązań umożliwiających łączenie się z serwerem z poziomu języka JavaScript. Jednym z tego typu rozwiązań jest Seam Remoting Framework, będący częścią opisanego wyżej szkieletu programistycznego JBoss Seam [45].

Do budowy przyjaznego interfejsu użytkownika w swojej aplikacji, korzystałem z możliwości oferowanych przez bibliotekę Ajax4JSF oraz Seam Remoting Framework.

3.2.2. Google Maps

Celem umożliwienia organizatorowi podania miejsca spotkania, system udostępnia możliwość zaznaczenia tego miejsca na mapie, która później pojawi się na stronie spotkania. Implementacja tej funkcjonalności wykorzystuje udostępniony przez firmę Google interfejs do popularnego serwisu Google Maps, pozwalający umieścić w swojej aplikacji mapę dowolnego obszaru i oprogramować jej zachowanie. Użycie interfejsu Google Maps API polega na dodaniu odpowiedniego kodu HTML i JavaScript do widoku aplikacji [15].

3.3. Komponent kalendarza

Jedną z najważniejszych części mojej aplikacji jest kalendarz, na którym użytkownicy mogą zaznaczać przedziały czasowe. Przed rozpoczęciem implementacji systemu rozważałem napisanie własnego komponentu kalendarza lub skorzystanie z któregoś z istniejących.

Istniejące otwarte implementacje kalendarza, wyświetlanego na stronie internetowej i napisane w języku Java, to komponent Schedule, będący częścią biblioteki MyFaces Tomahawk, kalendarz Bedework oraz Simile Timeline.

[3] Ściśle rzecz biorąc, AJAX nie jest technologią, a raczej techniką programistyczną, na którą składa się wiele technologii (zobacz [13]). Dla uproszczenia rozważań, będziemy jednak używać sformuowania „technologia AJAX".

3.3.1. Bedework

Bedework [4] jest rozbudowanym komponentem, pomyślanym początkowo jako kalendarz dla uczelni wyższych. Współpracuje on z wieloma formatami pozwalającymi na współdzielenie kalendarzy takimi jak iCal i CalDAV (patrz podrozdział 1.2.2). Bedework jest rozwijany na Rensselaer Polytechnic Institute.

3.3.2. Simile Timeline

Komponent Timeline [38], właściwie nie jest kalendarzem, ale osią czasową, do której, za pomocą prostego interfejsu (plik w formacie XML), można dodawać zdarzenia. Jest on częścią większego projektu Simile, prowadzonego na Massachusetts Institute of Technology, stawiającego sobie za cel dostarczenie narzędzi do zarządzania i wizualizacji zbiorów danych.

3.3.3. Tomahawk Schedule

Wykonany w technologii JSF komponent Schedule z biblioteki MyFaces Tomahawk [36], okazał się najlepiej przystosowanym do moich potrzeb. Oferuje on możliwość wyświetlania wydarzeń na kalendarzu w widoku dziennym, tygodniowym oraz miesięcznym. Dane o wydarzeniach pobierane są ze zwykłego obiektu Javy.

Żaden z rozważanych przeze mnie komponentów, łącznie z Tomahawk Schedule, nie posiadał jednak możliwości dodawania terminów poprzez zaznaczanie ich na kalendarzu. W celu zapewnienia takiej funkcjonalności, rozszerzyłem komponent z biblioteki Tomahawk, tworząc jego własną wersję, przystosowaną do potrzeb mojej aplikacji.

3.4. Baza danych

Jak wspomniano w podrozdziale 3.1.1, dzięki zastosowaniu technologii JavaPersistence API, uzyskujemy niezależność od używanej bazy danych. Końcowy projekt został skonfigurowany do pracy z bazą danych MySQL w wersji 5.0 [37]. Projekt był też testowany z użyciem „lekkiej" bazy danych HSQLDB [17].

Celem zapewnienia obsługi wielu języków, w tym języka polskiego, zarówno baza danych, jak i serwer aplikacyjny, zostały skonfigurowane do pracy w kodowaniu UTF-8.

3.5. Testowanie

Do testowania biblioteki IntervalLib (zobacz podrozdział 4.4) zostało wykorzystane narzędzie TestNG [39]. Narzędzie to umożliwia pisanie i uruchamianie różnego rodzaju testów (od testów jednostkowych po testy integracyjne) dla programów w języku Java. TestNG rozszerza funkcjonalność innego, podobnego narzędzia o nazwie JUnit [27].

Rozdział 4

Projekt i implementacja

4.1. Struktura projektu

W skład projektu wchodzą następujące katalogi:

- *src* – kod źródłowy napisany w języku Java,
- *lib* – zewnętrzne biblioteki,
- *resources* – pliki konfiguracyjne,
- *test* – konfiguracja testów,
- *view* – pliki definiujące widok.

Kod źródłowy został podzielony na pakiety:

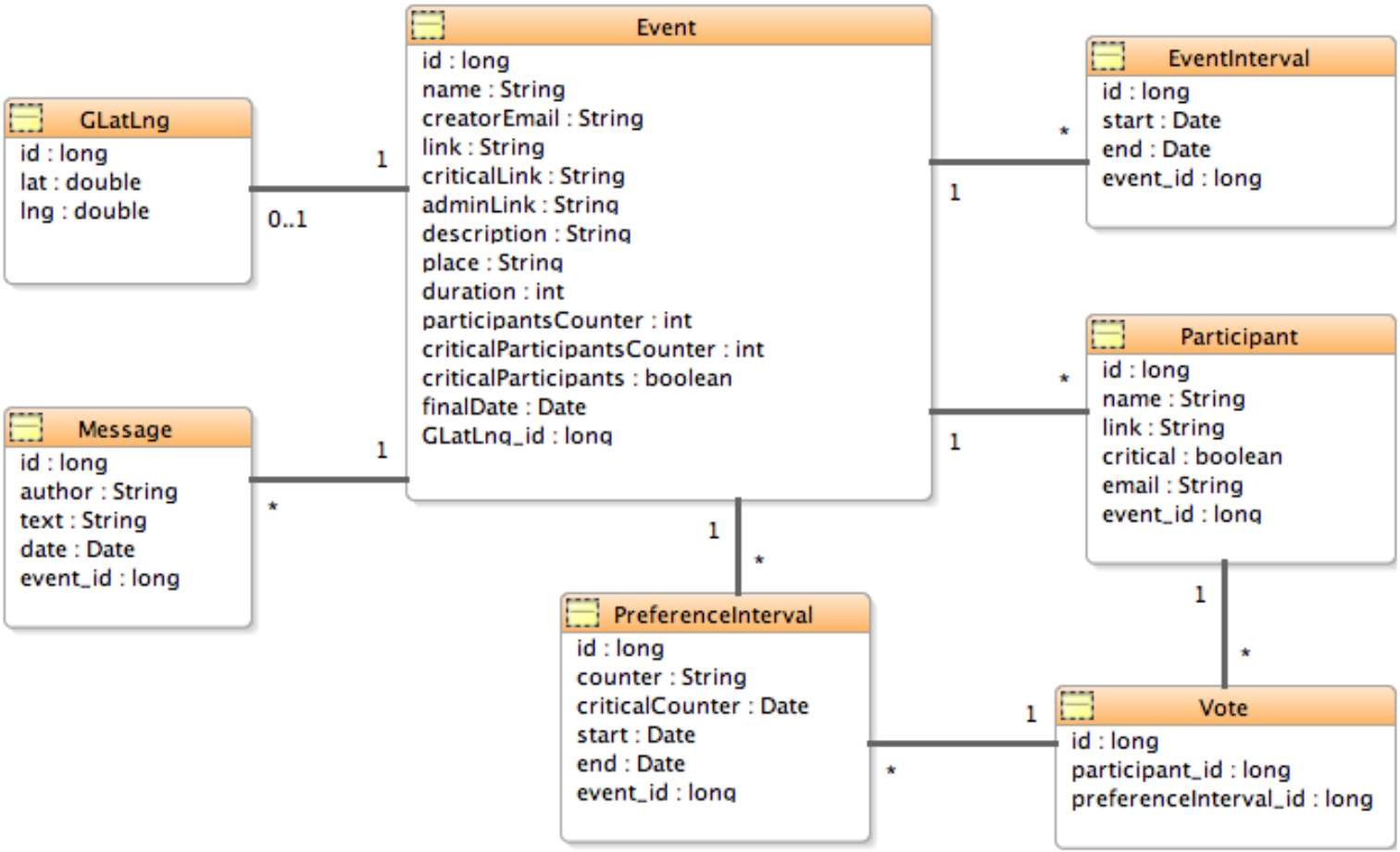
- *pl.jutroniemoge.entities* – model,
- *pl.jutroniemoge.actions* oraz *pl.jutroniemoge.services* – logika biznesowa,
- *pl.jutroniemoge.intervals* – biblioteka IntervalLib,
- *pl.jutroniemoge.utils* – kod wspólny dla pozostałych części aplikacji,
- *org.apache.myfaces.custom.schedule* i *pl.jutroniemoge.schedule* – przedefiniowany komponent Schedule z biblioteki Tomahawk.

W głównym katalogu projektu został umieszczony skrypt pozwalający zbudować projekt oraz plik *license.txt* opisujący warunki licencji na poszczególne części projektu oraz wykorzystywane biblioteki.

W następnych podrozdziałach dokładniej omawiam wymienione części projektu oraz role i zawartość poszczególnym pakietów.

4.2. Model

Od strony kodu źródłowego model jest reprezentowany jako zwykłe obiekty Javy (POJO). Obiekty, których zawartość ma być przechowywana w bazie danych, są encjami JavaPersistence API.



Rysunek 4.1: Schemat bazy danych

4.2.1. Schemat bazy danych

Rysunek 4.1 prezentuje schemat bazy danych. Dane przechowywane w tabelach oraz role poszczególnych pól są następujące:

- Event – reprezentuje pojedyncze spotkanie
  - id – klucz,
  - name – nazwa spotkania,
  - creatorEmail – adres e-mail organizatora spotkania,
  - link – unikatowy identyfikator spotkania, będący częścią odnośnika do spotkania,
  - criticalLink – unikatowy identyfikator dołączany do odnośnika dla uczestników krytycznych (patrz podrozdział 2.2),
  - adminLink – unikatowy identyfikator dołączany do odnośnika administracyjnego (patrz podrozdział 2.6),
  - description – opis spotkania
  - place – opis miejsca spotkania,
  - participantsCounter – liczba uczestników spotkania, którzy złożyli swoje preferencje. Ten oraz następny licznik zawierają informacje redundantne. (Można policzyć uczestników spotkania w tabeli Participant i otrzymać wartość licznika.) Są one utrzymywane z przyczyn wydajnościowych.
  - criticalParticipantsCounter – liczba uczestników krytycznych, którzy złożyli swoje preferencje,
  - criticalParticipants – informacja o tym, czy w spotkaniu biorą udział uczestnicy krytyczni,
  - finalDate – ostateczna data spotkania,
  - GLatLng_id – dowiązanie do współrzędnych geograficznych miejsca spotkania.
- Participant – uczestnik spotkania
  - id – klucz,
  - name – imię lub pseudonim podany przez uczestnika przy wejściu na stronę spotkania,
  - link – unikatowy identyfikator dołączany do prywatnego odnośnika pozwalającego zmienić swoje preferencje,
  - critical – informacja, czy uczestnik jest uczestnikiem krytycznym,
  - email – adres e-mail podany przy wejściu na stronę spotkania,
  - event_id – dowiązanie do spotkania.
- Vote – pojedyncza preferencja dotycząca jednego uczestnika i jednego przedziału czasowego
  - id – klucz,
  - participant_id – dowiązanie do uczestnika, który złożył daną preferencję,
  - preferenceInterval_id – dowiązanie do przedziału czasowego związanego z daną preferencją.

- EventInterval – przedział czasowy, w którym może odbyć się spotkanie
  - id – klucz,
  - start – początek przedziału,
  - end – koniec przedziału,
  - event_id – dowiązanie do spotkania.
- PreferenceInterval – przedział czasowy powiązany z preferencją
  - id – klucz,
  - counter – licznik uczestników, którzy zadeklarowali możliwość przybycia na spotkanie dokładnie w tym terminie (redundantny celem optymalizacji wydajności),
  - criticalCounter – j.w. dla uczestników krytycznych,
  - start – początek przedziału,
  - end – koniec przedziału,
  - event_id – dowiązanie do spotkania (redundantne, ale ustalenie spotkania, do którego należy dany przedział, wymagałoby wykonania potrójnego złączenia).
- Message – publiczna wiadomość wysłana przez uczestnika spotkania
  - id – klucz,
  - author – autor wiadomości. Jeśli uczestnik zrezygnuje ze spotkania lub zostanie z niego usunięty, to wysłane przez niego wiadomości pozostają widoczne. Dlatego w tym polu przechowywana jest nazwa użytkownika, a nie jego id.
  - text – treść wiadomości,
  - date – data wysłania wiadomości,
  - event_id – dowiązanie do spotkania.
- GLatLng – współrzędne geograficzne miejsca spotkania (porównaj [15])
  - id – klucz,
  - lat – szerokość geograficzna w stopniach,
  - lng – długość geograficzna w stopniach.

Należy wspomnieć, iż warstwa modelu zawiera także szereg klas pomocniczych, które nie są przechowywane w bazie danych. Obiekty tych klas są odpowiedzialne za konwersję wyświetlanych wartości, walidację oraz wybór języka.

4.2.2. Wydajność operacji na interwałach

Przy projektowaniu bazy danych główny nacisk został położony na zapewnienie wydajności operacji związanych z przedziałami czasowymi. Dla każdego spotkania przechowywane są dwa różnego rodzaju zbiory interwałów. Pierwszy z nich (elementy EvenInterval) to lista przedziałów czasowych, w których może odbyć się spotkanie. Lista ta stanowi zbiór prosty interwałów (zobacz podpunkt 4.4.1) zarządzany przez bibliotekę IntervalLib.

Na drugi ze zbiorów składają się przedziały czasowe odpowiadające preferencjom uczestników spotkania (elementy PreferenceInterval). Zbiór ten jest reprezentowany jako zbiór ważony

(zobacz podpunkt 4.4.2) w bibliotece IntervalLib. Reprezentacja przechowująca sumy preferencji poszczególnych uczestników została tutaj wprowadzona celem przyśpieszenia obliczania poziomu popularności poszczególnych przedziałów czasowych. Z kolei zapytanie o przedziały (PreferenceInterval) dla konkretnego spotkania (Event) będzie jedną z najczęściej wykonywanych operacji, stąd pomysł przechowywania nadmiarowego dowiązania do spotkania w elementach PreferenceInterval.

4.3. Logika biznesowa

Logika aplikacji odpowiada za wykonywanie akcji użytkownika oraz za dostarczanie danych dla modelu. Operuje ona na obiektach modelu i jest w pełni odseparowana od widoku. Szkielet programistyczny Seam odpowiada za wywołanie odpowiednich akcji logiki po zaistnieniu zdarzeń po stronie widoku lub w momencie gdy warstwie widoku są potrzebne nowe dane do wyświetlenia.

Odseparowanie widoku i logiki najłatwiej opisać korzystając z abstrakcyjnego pojęcia przestrzeni obiektów. Widok pobiera dane z należących do modelu obiektów będących w tej przestrzeni. Logika jest odpowiedzialna za umieszczanie odpowiednich obiektów (z odpowiednimi danymi) w przestrzeni. Wykonanie akcji użytkownika zwykle polega na dodaniu, usunięciu lub zmianie wartości obiektu znajdującego się w przestrzeni obiektów. W rzeczywistości jest wiele przestrzeni o różnych zakresach widoczności (m. in. strony, konwersacji, sesji).

Przykładowy przebieg sterowania według opisanego schematu ilustruje diagram sekwencji na rysunku 4.2. Przedstawia on przebieg sterowania dla akcji użytkownika związanych z utworzeniem spotkania. Akcje, zainicjowane przez aktora (organizator) po stronie widoku, są obsługiwane przez Seama. Ich wykonanie jest delegowane do komponentów zawierających logikę biznesową, w tym przypadku do obiektów klas CreateEvent, AddEventIntervalAction i SaveEventAction, który korzysta dodatkowo z serwisu LinkService. Elementy warstwy logiki operują na obiektach modelu (Event oraz EventInterval). Operacje związane z działaniami na zbiorze odcinków czasowych są delegowane do obiektu IntervalList, będącego elementem biblioteki IntervalLib. Na diagramie celowo umieściłem obok klas obiektów ich nazwy, ponieważ widok odwołuje się do obiektów pozostałych warstw właśnie przez nazwy.

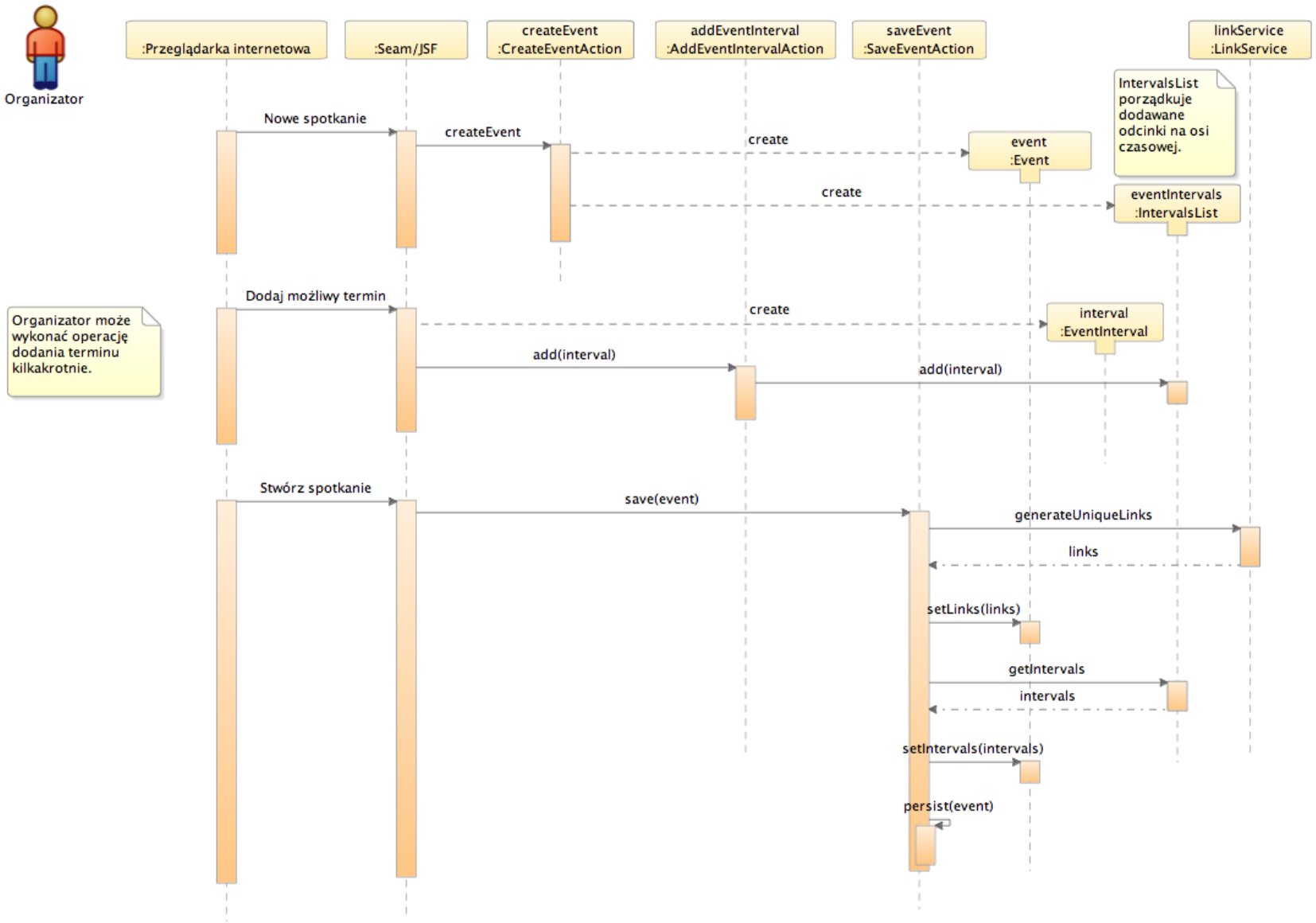
Warstwa logiki została zaimplementowana z użyciem technologii Enterprise JavaBeans 3.0 (patrz podrozdział 3.1.1). Część komponentów warstwy logiki jest wołana w sposób asynchroniczny (patrz 3.2), jednak szczegóły implementacji tych komponentów nie zależą od sposobu wywoływania ich metod.

Elementy logiki znajdują się w następujących podpakietach:

- *pl.jutroniemoge.actions.admin* – kontrolery akcji użytkownika tworzącego i administrującego spotkaniem,
- *pl.jutroniemoge.actions.participant* – kontrolery akcji użytkownika będącego uczestnikiem spotkania,
- *pl.jutroniemoge.services* – serwisy dostarczające dane niezależne od użytkownika.

4.3.1. Kontrolery akcji

Ogólną zasadą, której trzymałem się tworząc aplikację, było to, że każdej czynności użytkownika odpowiada jeden komponent EJB3. Jeśli dana czynność wymaga kilku akcji ze strony



Rysunek 4.2: Utworzenie spotkania - diagram sekwencji

użytkownika, to obsługa tych akcji może być zgrupowana w jednym lub wielu komponentach EJB3. Każdy komponent składa się z interfejsu oraz klasy implementującej ten interfejs[1].

Pełna lista kontrolerów akcji:

- Kontrolery akcji organizatora spotkania
  - CreateEvent[2] – utwórz spotkanie (bez zapisywania go do bazy danych),
  - SaveEvent – zapisz spotkanie w bazie danych,
  - DeleteEvent – usuń spotkanie. Usuwa kaskadowo wszystkie dane związane ze spotkaniem oraz jego uczestnikami,
  - UpdateEvent – uaktualnij parametry spotkania,
  - UpdateEventIntervals – uaktualnij listę terminów, w których może się odbyć spotkanie,
  - AddEventInterval – dodaj interwał, w którym może się odbyć spotkanie. Deleguje działania na odcinkach czasowych do biblioteki IntervalLib,
  - RemoveEventInterval – usuń interwał z listy terminów, w których może się odbyć spotkanie. Deleguje działania na odcinkach czasowych do biblioteki IntervalLib,
  - SwitchCriticalParticipant – zmień status uczestnika ze zwykłego na uczestnika krytycznego lub odwrotnie. Zmiana pociąga za sobą zmianę sposobu liczenia preferencji dla danego uczestnika,
  - AdminRemoveParticipant – usuń uczestnika. Usuwa także wszystkie preferencje złożone przez daną osobę,
  - FinishVoting – zakończ głosowanie (zobacz podrozdział 2.4).
- Kontrolery akcji uczestnika spotkania
  - AddParticipant – utwórz nowego uczestnika (uczestnik jest od razu zapisywany do bazy danych),
  - RemoveParticipant – rezygnacja ze spotkania. Wszystkie dane o uczestniku, wraz z jego preferencjami, zostaną usunięte,
  - AddVote – dodaj preferencję. Deleguje działania na odcinkach czasowych do biblioteki IntervalLib,
  - RemoveVote – usuń wcześniej złożoną preferencję. Deleguje działania na odcinkach czasowych do biblioteki IntervalLib,
  - SaveVotes – kończy składanie preferencji po pierwszym wejściu uczestnika na stronę spotkania i wysyła mu list elektroniczny z jego prywatnym odnośnikiem,
  - UpdateVotes – uaktualnia złożone preferencje,
  - AddMessage – umożliwia umieszczenie publicznej wiadomości na stronie spotkania.

[1] To, że każdy komponent EJB3 musi posiadać interfejs, jest wymogiem specyfikacji [11]. Być może ten warunek zniknie w następnej wersji standardu (zobacz [6]).

[2] Podane nazwy odnoszą się do interfejsów. Obiekty implementujące te interfejsy mają nazwy takie same jak interfejsy z sufiksem „Action” (lub „Bean” dla serwisów).

4.3.2. Serwisy

Serwisy stanowią część logiki, która nie jest bezpośrednio związana z konkretnymi akcjami użytkownika. Odnajdują one lub generują na żądanie obiekty potrzebne warstwie widoku. Oto lista wszystkich serwisów i ich role:

- EventService – odnajduje obiekt reprezentujący spotkanie na podstawie odnośnika z unikatowym identyfikatorem spotkania,
- ParticipantService – j.w. dla uczestnika,
- BestIntervalListService – dostarcza listę terminów uporządkowaną według popularności. Korzysta z biblioteki IntervalLib,
- GoogleMapsConnectorService – obsługa komunikacji z Google Maps API,
- UpdaterService – odpowiada za okresowe uaktualnianie danych wyświetlanych użytkownikowi,
- LinkService – generowanie unikatowych odnośników.

4.3.3. Uaktualnianie danych i współbieżność

System umożliwia automatyczne odświeżanie danych prezentowanych użytkownikom na stronie spotkania. Innymi słowy, oglądając stronę spotkania jego uczestnik będzie widział (z niewielkim opóźnieniem) efekty zmian wprowadzanych równocześnie przez innych uczestników spotkania bez potrzeby przeładowywania strony. Dotyczy to zarówno preferencji terminów, jak i wiadomości w dyskusji.

Częstotliwość uaktualnień jest konfigurowalna, nie może być jednak zbyt wysoka z przyczyn wydajnościowych. Może to oznaczać na przykład uaktualnianie wyświetlanych danych co minutę.

Ponadto system jest odporny na równoczesną zmianę preferencji przez wiele osób, także gdy osoby te korzystają z tej samej przeglądarki na tym samym komputerze. Obsługiwane są też sytuacje takie jak skreślenie z listy uczestników osoby, która właśnie składa lub zmienia swoje preferencje. W celu zapewnienia atomowości operacji oraz spójności danych przy równoczesnym dostępie do systemu przez wiele osób, aplikacja wykorzystuje mechanizm transakcji.

4.4. Biblioteka IntervalLib

Biblioteka IntervalLib odpowiada za operacje na zbiorach odcinków. Odcinki znajdujące się w tych zbiorach są uporządkowane liniowo. Wyróżniam dwa typy zbiorów odcinków – prosty i ważony. Zostaną one szczegółowo omówione w dalszej części tego punktu.

Zbiory te operują na dowolnych, domkniętych odcinkach, których końce są liczbami całkowitymi. W praktyce zostały ona wykorzystane do działań na interwałach czasowych, dlatego w dalszej części rozważań będziemy się posługiwali przykładami dotyczącymi operacji na przedziałach czasowych.

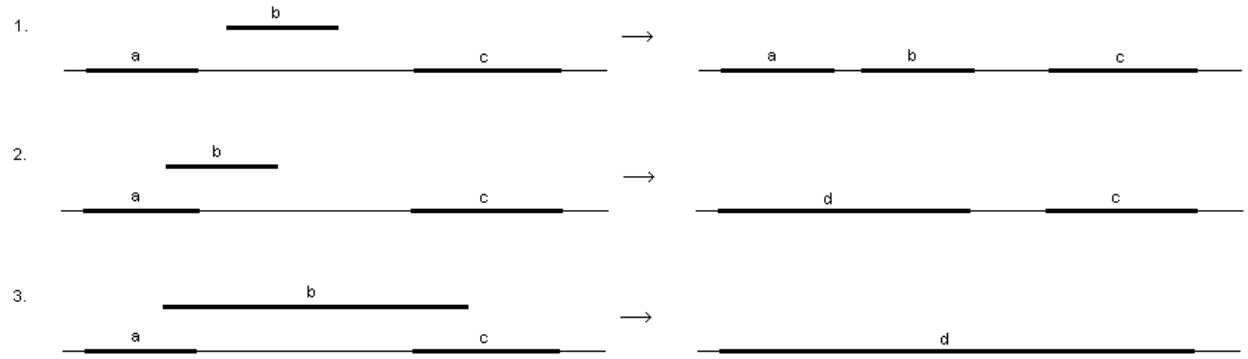
Biblioteka IntervalLib stanowi jedną z najważniejszych części całej aplikacji. Wykonuje ona działania na zbiorze możliwych terminów spotkania oraz na liście terminów, w których uczestnik może przybyć na spotkanie. Odpowiada za utrzymywanie listy sumarycznych preferencji oraz za generowanie listy potencjalnych terminów wraz z informacją o popularności poszczególnych przedziałów czasowych.

4.4.1. Zbiór prosty

Operacje na zbiorze prostym obejmują:

- Utworzenie pustego zbioru,
- Dodanie odcinka,
- Usunięcie odcinka znajdującego się w zbiorze,
- Sprawdzenie, czy dany odcinek znajduje się w zbiorze,
- Zapytanie o liczbę odcinków w zbiorze. W szczególności zapytanie o to, czy zbiór jest pusty.

Rysunek 4.3 prezentuje operację dodawania odcinka do zbioru. Jeśli dodawany odcinek (b) zachodzi, w części lub w całości, na odcinki będące już w zbiorze (a i c), to taka operacja może połączyć wiele odcinków w jeden (d).



Rysunek 4.3: Operacja dodawania odcinka do zbioru prostego

Zbiór prosty jest wykorzystywany przy tworzeniu listy terminów spotkania oraz przy składaniu preferencji przez uczestnika spoktania. W obu przypadkach implementacja zbioru prostego zapewnia wykonywanie operacji dodawania i usuwania przedziałów czasowych zgodnie z powyższym opisem. Dodawanie odcinka może powodować konieczność usunięcia istniejących już przedziałów i zastąpienia ich jednym, dłuższym. Tego typu zmiany są odzwierciedlane w widoku kalendarza oraz zapisywane w bazie danych.

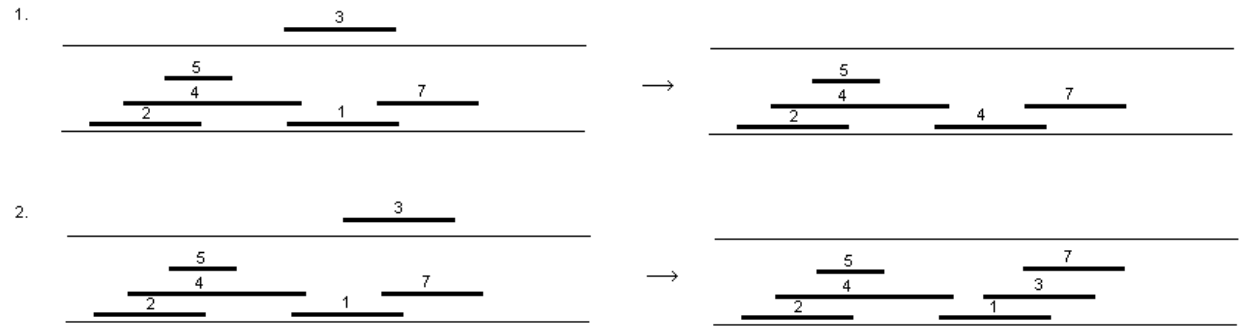
4.4.2. Zbiór ważony

W zbiorze ważonym z każdym odcinkiem związany jest licznik. Odcinki w zbiorze mogą na siebie nachodzić. Są one posortowane leksykograficznie według początków i końców.

Operacja dodawania odcinka w zbiorze ważonym została przedstawiona na rysunku 4.4. Zostały na nim zaprezentowane dwa przypadki dodawania odcinka z wagą trzy do zbioru składającego się z pięciu odcinków.

Jeśli w zbiorze istnieje odcinek o granicach takich samych jak dodawany (1), to licznik odcinka ze zbioru jest odpowiednio zwiększany. W przeciwnym wypadku (2) odcinek jest wstawiany do zbioru.

Zbiór ważony odpowiada za operacje na liście sumarycznych preferencji. Lista ta zawiera wszystkie przedziały, jakie zostały zgłoszone przez uczestników spotkania jako terminy, w



Rysunek 4.4: Operacja dodawania odcinka do zbioru złożonego

których mogą wziąć udział w spotkaniu. Z każdym przedziałem jest związana waga odpowiadająca liczbie uczestników, którzy zgłosili dany przedział.

W tym miejscu warto opisać pokrótce operacje wykonywane przez bibliotekę IntervalLib w momencie dodawania przez uczestnika nowego przedziału czasowego do listy swoich preferencji. System najpierw wstawia przedział do zbioru prostego przechowującego listę preferencji danego uczestnika. W ten sposób powstaje lista zmian, jakie zostały spowodowane dodaniem nowego interwału. Lista zmian obejmuje dodanie co najmniej jednego przedziału czasowego oraz, ewentualnie, usunięcie niektórych przedziałów. Zmiany z listy preferencji uczestnika są następnie uwzględniane w liście sumarycznej preferencji (poprzez wykonanie odpowiednich operacji dodawania i usuwania interwałów), przechowywanej w zbiorze ważonym. Podobnie wygląda operacja usunięcia przez uczestnika terminu z listy preferencji.

Ponadto zbiór ważony udostępnia operację wybrania ze zbioru wszystkich przedziałów o długości równej lub większej niż zadana i uporządkowanie ich według wag. W tej operacji brane są pod uwagę także przecięcia przedziałów znajdujących się w zbiorze. Jest to operacja wykorzystywana przez aplikację do obliczenia poziomu popularności poszczególnych przedziałów czasowych. Muszą być brane pod uwagę przecięcia odcinków, ponieważ jeśli np. 3 osoby zadeklarowały się, że mogą uczestniczyć w spotkaniu między 12.00 a 15.00, natomiast 2 inne osoby mają czas od godziny 13.30 do 16.30, to, przedział czasowy 13.30–15.00 ma największą popularność, gdyż w tym czasie w spotkaniu może wziąć udział 5 osób.

Jeśli w spotkaniu biorą udział uczestnicy krytyczni, to wagi każdego przedziału składają się z dwóch liczników zawierających liczbę wszystkich uczestników, którzy zgłosili dany przedział jako swoją preferencję oraz liczbę uczestników krytycznych, którzy go zadeklarowali. Przy ustalaniu popularności terminów stosowany porządek na wagach jest porządkiem leksykograficznym – najpierw porównywana jest liczba uczestników krytycznych dla danego przedziału, a dopiero w drugiej kolejności liczba wszystkich uczestników.

4.4.3. Implementacja

W trakcie relatywnie długich poszukiwań bibliotek udostępniających operacje na interwałach udało mi się znaleźć jedynie projekt JScience [25], w którego dokumentacji opisana została klasa IntervalList implementująca zbiór prosty. Niestety klasa ta należała do eksperymentalnej części projektu [26]. Jak wyraźnie zaznaczono w kodzie źródłowym klasy, dokończenie implementacji wymagało jeszcze wiele pracy. Napisane przeze mnie testy potwierdziły to

stwierdzenie. Nie udało mi się także znaleźć otwartego projektu, który próbowałby zmierzyć się z implementacją zbioru ważonego lub pokrewnym zagadnieniem. W związku z tym postanowiłem stworzyć własną implementację obu zbiorów.

Implementacja biblioteki IntervalLib okazała się zadaniem wymagającym dużo precyzji, choćby z racji różnych konfiguracji, w których może się znaleźć dodawany lub usuwany odcinek. Osobny problem stanowił sposób obliczania popularności poszczególnych terminów, a ujmując to w języku odcinków, algorytm obliczający wagi poszczególnych przecięć odcinków znajdujących się z zbiorze ważonym. Reprezentacja zbioru ważonego została tutaj dobrana nieprzypadkowo. Jak już wspomniano, niezbędne jest wzięcie pod uwagę także przecięć odcinków znajdujących się w zbiorze. Zauważmy jednak, że dla każdego odcinka czasu ze zbioru wystarczy przeanalizować przecięcia rozpoczynające się w tym odcinku. Ponadto jedną z kluczowych kwestii jest to, żeby nie wliczać dwukrotnie różnych preferencji pochodzących od tej samej osoby. W opisanej reprezentacji zbioru ważonego nie zdarzy się sytuacja, w której dwa przecinające się przedziały dotyczą głosu pochodzącego od tej samej osoby.

Implementacja biblioteki znajduje się w pakiecie *pl.jutroniemoge.interval*. Zbiór prosty reprezentuje klasa IntervalList, natomiast zbiór złożony klasa CounterIntervalList.

4.4.4. Testy

Jako że biblioteka IntervalLib miała odpowiadać za wykonywanie najważniejszych dla całej aplikacji operacji, niezbędne było jej przetestowanie. Stworzyłem zestawy testów zarówno dla implementacji zbioru prostego, jak i zbioru ważonego.

Część testów to testy weryfikujące poprawność działania poszczególnych operacji na zbiorach, takich jak na przykład dodawanie odcinka znajdującego się w różnych konfiguracjach względem wcześniej wstawionych do zbioru odcinków. Testy sprawdzają także zachowanie implementacji zbiorów w sytuacjach skrajnych (np. zbiór pusty lub jednoelementowy).

Osobną grupę testów stanowią testy losowe. Testy te sprawdzają przemienność operacji dodawania i usuwania odcinków w zbiorze. Ich idea polega na wielokrotnym uruchomieniu testu, który losuje zestaw interwałów, tworzy dwa zbiory puste, a następnie wstawia wylosowane interwały do zbiorów (w różnych, losowych kolejnościach dla różnych zbiorów). Operacje dodawania mogą być przemieszane z operacjami usuwania pewnych, wcześniej ustalonych, odcinków ze zbioru. Na końcu sprawdzamy czy zbiory po wykonaniu wszystkich operacji są równe.

Opisane testy zostały umieszczone w podpakiecie *pl.jutroniemoge.intervals.test*.

4.5. Widok

Warstwa widoku została wykonana z użyciem technologii JSF i Facelets (patrz podrozdział 3.1.2). Na warstwę widoku składają się następujące pliki:

- szablon stosowany na wszystkich stronach aplikacji (template.xhtml),
- pliki w formacie XML (z rozszerzeniem .xhtml), definiujące poszczególne widoki i ich elementy. Definicje widoków stron dla organizatora spotkania umieszczono w podkatalogu *admin*, widoki stron używanych przez uczestników spotkania w podkatalogu *participant*, natomiast elementy wspólne w podkatalogu *common*.
- definicje treści wiadomości e-mail, wysyłanych do użytkowników systemu (w formacie XML) – podkatalog *email*,

- pliki zawierające kod w języku JavaScript – podkatalog *js*,
- kaskadowe arkusze stylów – podkatalog *css*,
- obrazki – podkatalog *img*.

Wszystkie wymienione pliki i podkatalogi znajdują się w katalogu *view* projektu.

4.5.1. Google Maps

O ile umieszczenie samej mapy z Google Maps na stronie internetowej jest dość prostą czynnością, to podłączenie interfejsu Google Maps do całości aplikacji wymaga użycia któregoś z mechanizmów integracji JavaScriptu z kodem aplikacji umieszczonym na serwerze. Komunikacja ta została zrealizowana z użyciem Seam Remoting Framework (patrz podrozdział 3.2). Jej implementacja składa się z modułu działającego po stronie klienta oraz odpowiednich obiektów umieszczonych na serwerze aplikacji.

Moduł obsługujący Google Maps po stronie klienta jest napisany w języku JavaScript. Jest on odpowiedzialny za wyświetlanie i ukrywanie mapy oraz przekazywanie informacji między komponentem a serwerem. Na stronie nowego spotkania (oraz stronie administracyjnej) moduł ten przesyła do serwera współrzędne geograficzne klikniętego przez użytkownika miejsca. Na stronie dla uczestnika spotkania odpowiedni kod JavaScript ściąga z serwera informację o miejscu spotkania i wyświetla na mapie.

Po stronie serwera za komunikację z Google Maps odpowiada wspomniany już serwis GoogleMapsConnectorService, który dostarcza metody do pobrania oraz zapisania danych wyznaczających miejsce spotkania.

4.5.2. Kalendarz

Komponent kalendarza został zbudowany na podstawie komponentu Schedule z biblioteki MyFaces Tomahawk [36]. W ramach zmian przedefiniowałem oryginalną klasę ScheduleDetailedDayRenderer. Dodałem generowanie kodu HTML i JavaScript odpowiadającego za obsługę odpowiednich zdarzeń (naciśnięcie, zwolnienie przycisku myszy, przeciąganie).

Zdarzenia te powodują wywołanie napisanego przeze mnie kodu JavaScript, który na bieżąco wyświetla zaznaczane przez użytkownika obszary, a po zakończeniu zaznaczania przesyła informację o zaznaczonym fragmencie do serwera. Komunikacja z serwerem, podobnie jak w przypadku Google Maps, jest zaimplementowana z użyciem Seam Remoting Framework (patrz podrozdział 3.2). Po stronie serwera zaznaczony przedział czasowy zostaje dodany do odpowiedniego zbioru. Następnie uaktualniany jest widok kalendarza.

Przy składaniu preferencji uczestnik może zaznaczać tylko terminy mieszczące się w granicach wyznaczonych przez listę terminów, w których może odbyć się spotkanie. Sprawdzanie tego warunku jest dokonywane na bieżąco na komputerze użytkownika, po wcześniejszym ściągnięciu z serwera listy terminów spotkania.

W ramach własnej implementacji komponentu kalendarza dodałem także swoją wersję klasy ScheduleEntryRenderer, która jest odpowiedzialna za renderowanie pojedynczych przedziałów czasowych. Za jej pomocą można m.in. zmieniać kolory i zawartość wyświetlanych przedziałów czasowych. Dodaje ona do kalendarza możliwość usuwania interwałów poprzez kliknięcie na nie.

Ponadto komponent Schedule w widoku tygodniowym wyświetlał dni robocze w pojedynczych kolumnach, natomiast sobota i niedziela jako dni wolne od pracy były umieszczane zawsze w jednej kolumnie. Na potrzeby aplikacji opcja ta została przeprogramowana, dzięki czemu wświetlane są pełne kolumny dla każdego dnia tygodnia.

Niezbędne okazło się też wprowadzenie niewielkich poprawek i dodanie plików konfiguracyjnych umożliwiających działanie biblioteki Tomahawk z Faceletami, biblioteką RichFaces oraz z implementacją referencyjną JSF zamiast dostarczanej przez Apache Software Foundation (zobacz podpunkt 3.1.2). Z racji różnic między przeglądarkami w obsłudze skryptów JavaScript, zdecydowałem się na zaimplementowanie komponentu kalendarza zgodnego z jedną przeglądarką (Mozilla Firefox [33] w wersji 2.0).

4.6. Pliki konfiguracyjne

Z ważniejszych plików konfiguracyjnych należy wymienić:

- Skrypt dla Apache Ant [2] *build.xml* służący do zbudowania projektu, osadzenia go na serwerze oraz przeprowadzenia testów,
- Pliki zawierające napisy do wyświetlenia na stronach WWW w języku polskim oraz angielskim: *jutroniemoge_messages_pl.properties* i *jutroniemoge_messages_en.properties*,
- Konfigurację połączenia z bazą danych: *jutroniemoge-prod-ds.xml* oraz *persistence-prod.xml*,
- Pliki konfiguracyjne dla serwera aplikacji: *web.xml*, *application.xml* oraz *ejb-jar.xml*,
- Konfigurację szkieletu programistycznego Seam: *components.xml* i *pages.xml*,
- Pliki JSF: *faces-config.xml*, *tomahawk.taglib.xml*.

  Wszystkie wymienione wyżej pliki, oprócz skryptu *build.xml*, znajdują się w katalogu *resources* i jego podkatalogach.

Rozdział 5

Interfejs użytkownika

W rozdziale tym opisano krok po kroku jak, korzystając z interfejsu aplikacji, zdefiniować spotkanie i administrować nim oraz jak złożyć preferencje. Opisy wymienionych czynności poparte są ilustracjami przedstawiającymi wygląd interfejsu.

5.1. Strona główna

Ze strony głównej systemu będą korzystały osoby chcące zdefiniować spotkanie oraz dowiedzieć się, co system ma do zaoferowania. Wygląd strony głównej został przedstawiony na rysunku 5.1. Jest ona dość prosta w budowie i zawiera krótki opis zachęcający do skorzystania z serwisu oraz przycisk pozwalający utworzyć nowe spotkanie.

Rysunek 5.1: Strona główna serwisu

Strona ta celowo nie opisuje wszystkich cech systemu, jak kalendarz oferujący natychmiastowy podgląd listy najpopularniejszych terminów, możliwość późniejszej zmiany preferencji czy definiowanie uczestników krytycznych. Osoby chcące dokładniej zapoznać się z funkcjonalnością serwisu przed jego pierwszym użyciem mogą znaleźć więcej informacji na stronie

do której prowadzi odnośnik „Dowiedz się więcej”.

5.2. Definiowanie spotkania

Spotkanie może stworzyć dowolny użytkownik, korzystając z przycisku „Utwórz spotkanie”, umieszczonego na stronie głównej. Nie jest wymagane zalogowanie się w serwisie.

Definiowanie spotkania przebiega w dwóch etapach. Najpierw organizator podaje podstawowe dane dotyczące spotkania, a następnie wybiera terminy, w których może się ono odbyć.

Pierwszy z wymienionych kroków, czyli wprowadzenie podstawowych danych o spotkaniu, przedstawia rysunek 5.2. Organizator proszony jest o wypełnienie zwykłego formularza, na którym podaje nazwę i długość spotkania oraz swój adres e-mail, na który zostanie mu przesłany odnośnik do spotkania. Dodatkowo organizator może dodać opis spotkania i jego miejsca oraz zadecydować, czy w spotkaniu będą brali udział uczestnicy krytyczni.

Dostępna jest także mapa, na której użytkownik może zaznaczyć miejsce spotkania. Na mapie można szukać konkretnych miejsc dzięki opcji „Idź do”, bądź przesuwać się po niej za pomocą myszy. Kliknięcie powoduje zaznaczenie miejsca spotkania w odpowiednim miejscu.

Kolejnym etapem definiowania spotkania jest podanie listy terminów, w których może się ono odbyć (rys. 5.3). Użytkownik ma do dyspozycji dwie metody wprowadzania przedziałów czasowych. Może wpisywać lub wybierać daty w formularzu albo zaznaczać je na kalendarzu.

Dodanie przedziału, który częściowo lub w całości pokrywa się z istniejącym, powoduje połączenie obu przedziałów w jeden. Istnieje także możliwość usunięcia przedziału. Należy w tym celu wybrać przedział do usunięcia z listy wszystkich przedziałów lub kliknąć „usuń” na przedziale na kalendarzu.

Kalendarz pozwala na przeglądanie wprowadzonych przedziałów czasowych w widoku tygodniowym. Dla wygody przeglądania można zawęzić godziny dnia, które są prezentowane na kalendarzu (np. wybrać tylko godziny pracy albo wieczorne). Do nawigacji, tj. wyboru tygodnia, miesiąca czy roku, służy mniejszy kalendarz umieszczony ponad kalendarzem głównym.

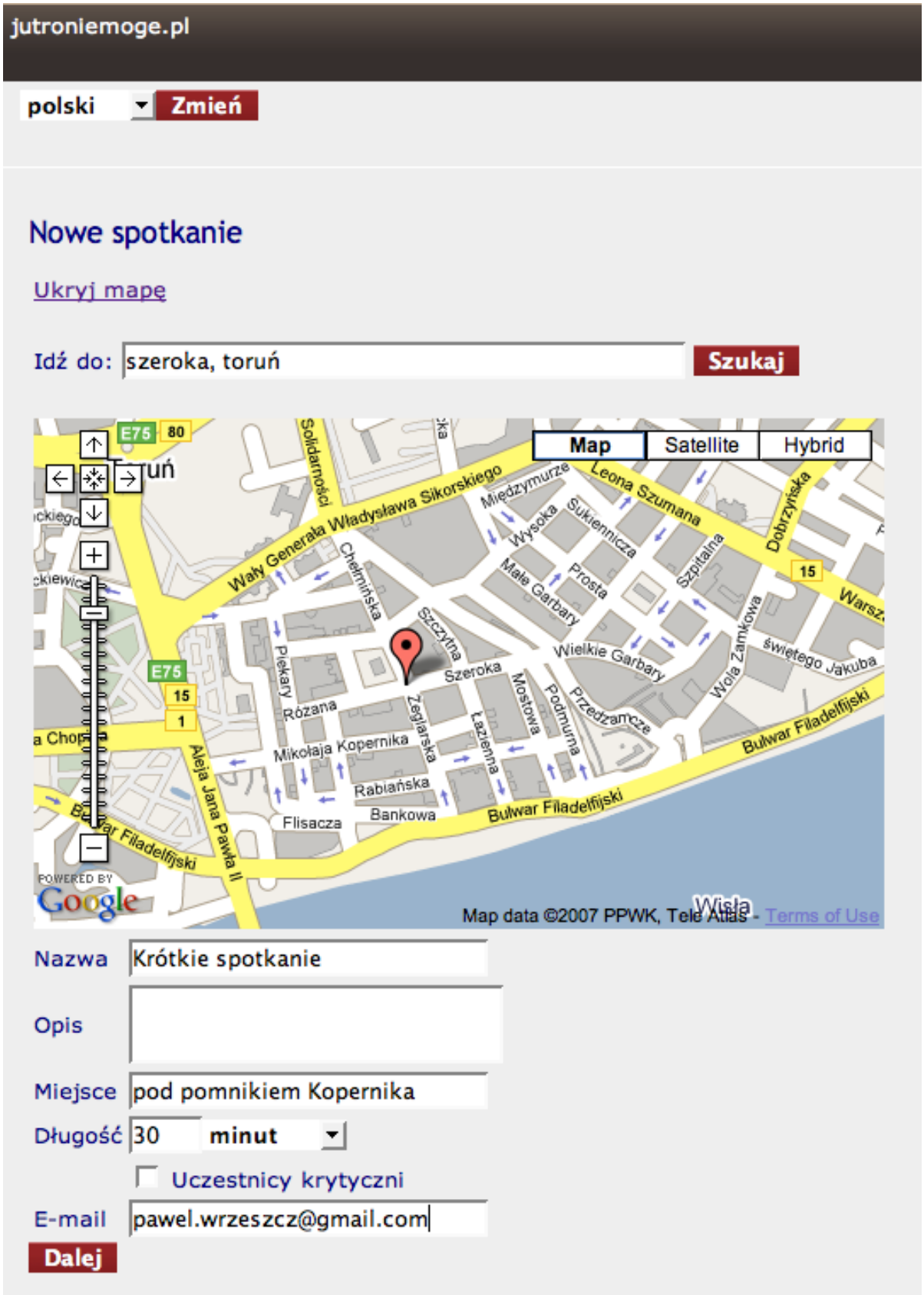
Po zdefiniowaniu spotkania system wyświetla organizatorowi stronę potwierdzającą utworzenie spotkania, zawierającą odnośniki do spotkania (dla uczestników, ew. dla uczestników krytycznych oraz odnośnik administracyjny). Informacje te zostają też przesłane organizatorowi listem elektronicznym na podany przez niego wcześniej adres.

5.3. Składanie preferencji

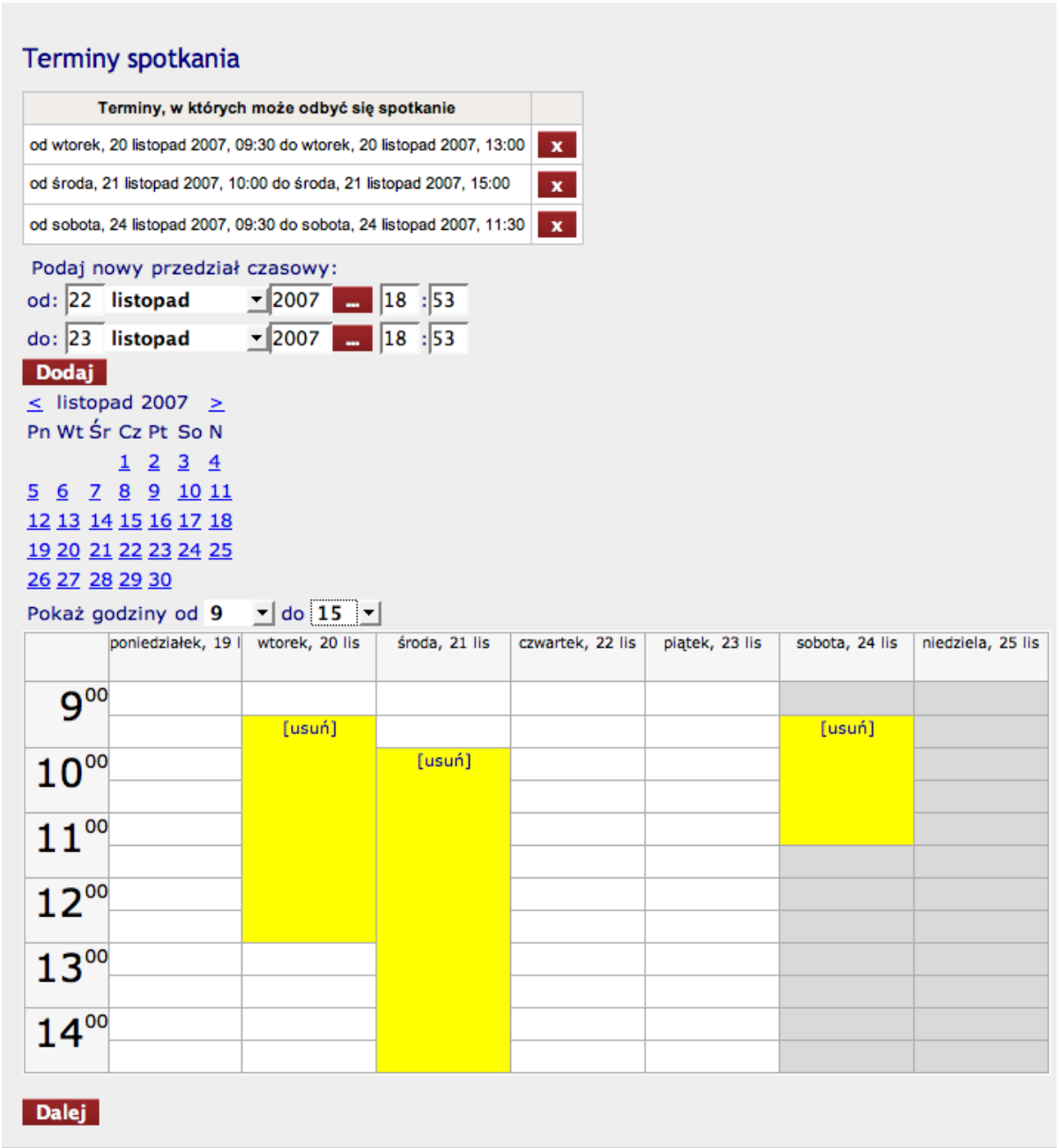
Po wejściu na stronę spotkania można przeglądać listę osób, które złożyły już swoje preferencje oraz listę możliwych terminów spotkania, uporządkowanych według ich popularności. Złożenie własnych preferencji wymaga zarejestrowania się.

Aby zarejestrować się jako uczestnik i tym samym uzyskać możliwość złożenia swoich preferencji, użytkownik musi podać jedynie swoje imię lub pseudonim. Może też podać swój adres e-mail, na który zostanie mu przysłany jego prywatny odnośnik do spotkania. Podanie swojego imienia pozwala także na wysyłanie publicznych wiadomości. Rysunek 5.4 przedstawia widok strony spotkania po zarejestrowaniu się uczestnika o imieniu Andrzej.

Użytkownik składa preferencje tworząc listę terminów, w których może uczestniczyć w spotkaniu. Przedziały czasowe można, podobnie jak w wypadku definiowania spotkania, podawać wprowadzając daty w formularzu lub zaznaczając je na kalendarzu.

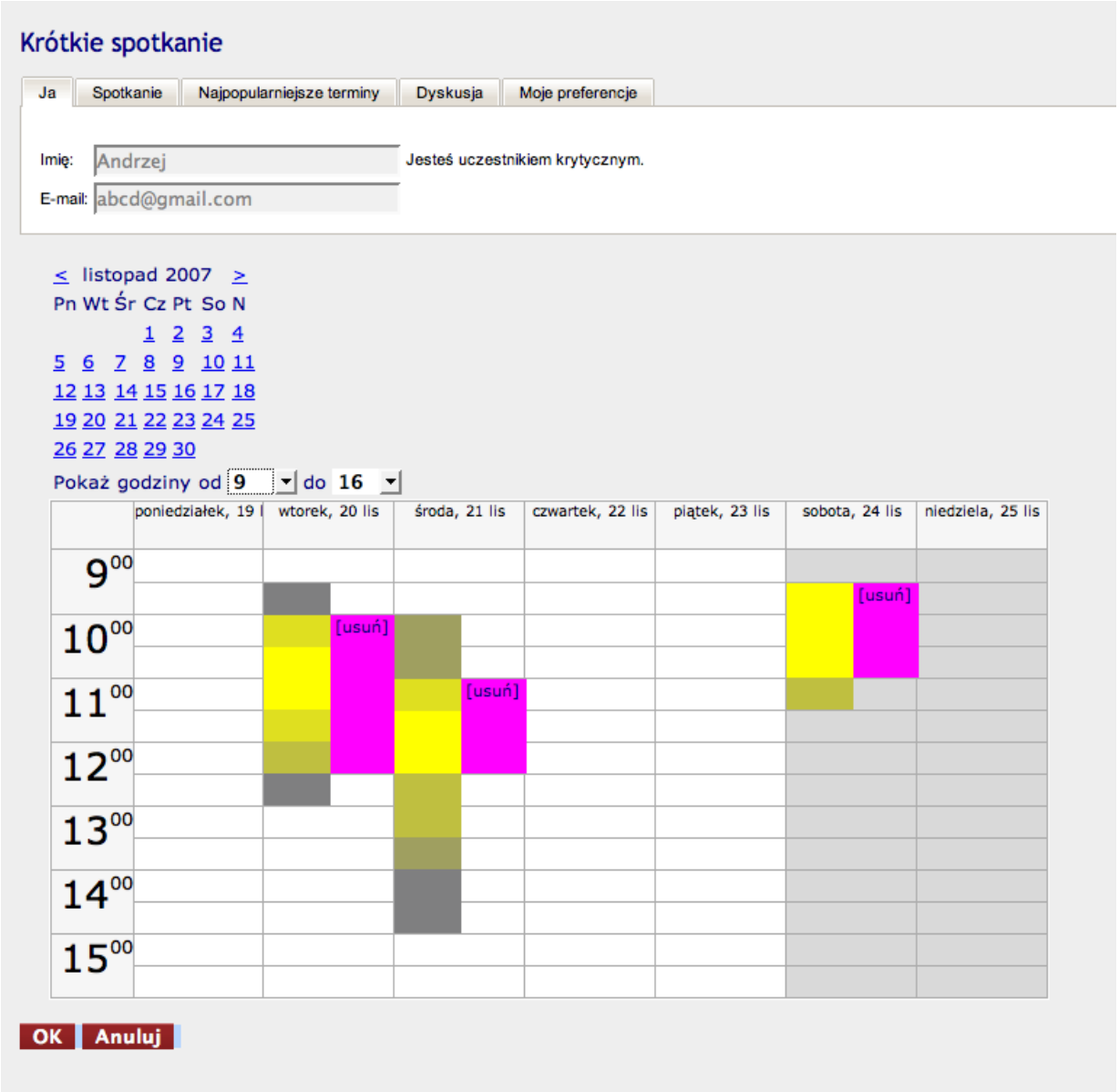


Rysunek 5.2: Definiowanie spotkania



Rysunek 5.3: Definiowanie listy terminów spotkania

Kalendarz na stronie spotkania zawiera dwie osobne listy terminów – listę możliwych terminów spotkania oraz listę preferencji danego uczestnika.



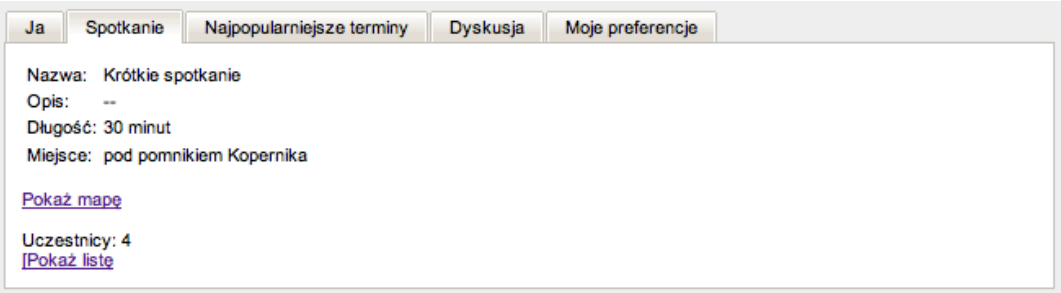
Rysunek 5.4: Składanie preferencji

Elementy listy terminów spotkania (sumaryczne preferencje) są umieszczane po lewej stronie kolumn zawierających poszczególne dni. Intensywność koloru obrazuje popularność danego terminu. Po najechaniu myszką na wybrany termin można sprawdzić, jaka część spośród wszystkich uczestników mogłaby uczestniczyć w spotkaniu w tym terminie. Kolorem szarym oznaczone są terminy, w których mogłoby odbyć się spotkanie, ale na razie nikt nie zgłosił możliwości uczestnictwa w tych terminach.

Preferencje czasowe uczestnika są pokazane po prawej stronie kolumn poszczególnych dni. Jest to też miejsce, w którym uczestnik może zaznaczać kolejne terminy. Aplikacja pozwala na zaznaczanie przedziałów czasowych zawierających preferencje tylko w zakresie możliwych terminów spotkania. Preferencje można także usuwać.

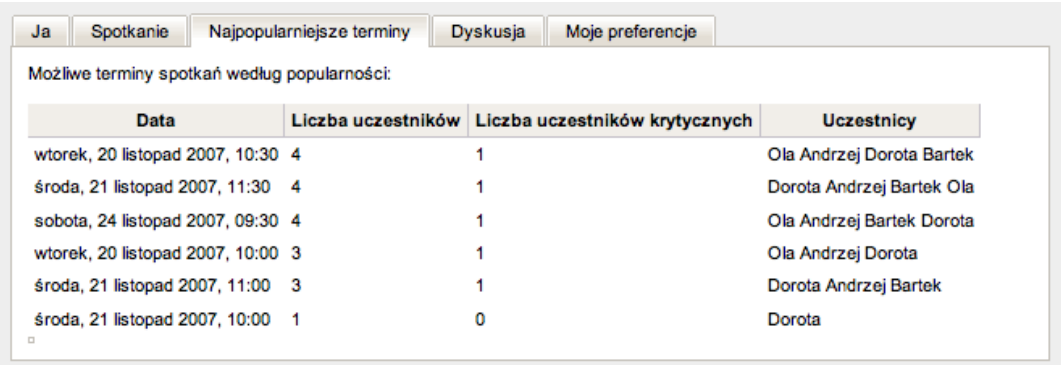
Sumaryczne preferencje są odświeżane po każdym dodaniu lub usunięciu przedziału czasowego przez uczestnika. Dzięki temu użytkownik ma na bieżąco wgląd w to, jaki globalny skutek przyniesie zmiana jego preferencji. Niezależnie od tego lista terminów spotkania oraz poziomy popularności poszczególnych terminów są periodycznie uaktualniane. W czasie pracy użytkownika popularność terminów może się zmieniać w miarę modyfikowania lub dodawania preferencji przez pozostałych uczestników spotkania i zmiany te będą odnotowywane na stronie, którą widzi uczestnik.

Dodatkowa funkcjonalność oferowana uczestnikowi na stronie spotkania zawarta jest w zakładkach na górze strony. Zakładka „Spotkanie" (rys. 5.5) umożliwia zapoznanie się z dodatkowymi informacjami dotyczącymi spotkania, takimi jak jego opis, długość czy miejsce. W zakładce dostępna jest mapka wskazująca miejsce spotkania. Można tutaj też znaleźć listę wszystkich uczestników spotkania.



Rysunek 5.5: Zakładka „Spotkanie"

Zakładka „Najpopularniejsze terminy" (rys. 5.6) zawiera uporządkowane według popularności terminy, w których mogłoby odbyć się spotkanie wraz z liczbą i listą uczestników, którzy mogliby wziąć udział w spotkaniu w poszczególnych terminach. Jeśli w spotkaniu mają wziąć udział uczestnicy krytyczni, to dodatkowa kolumna podaje liczbę uczestników krytycznych, którzy zadeklarowali gotowość wzięcia udziału w spotkaniu w danym terminie.

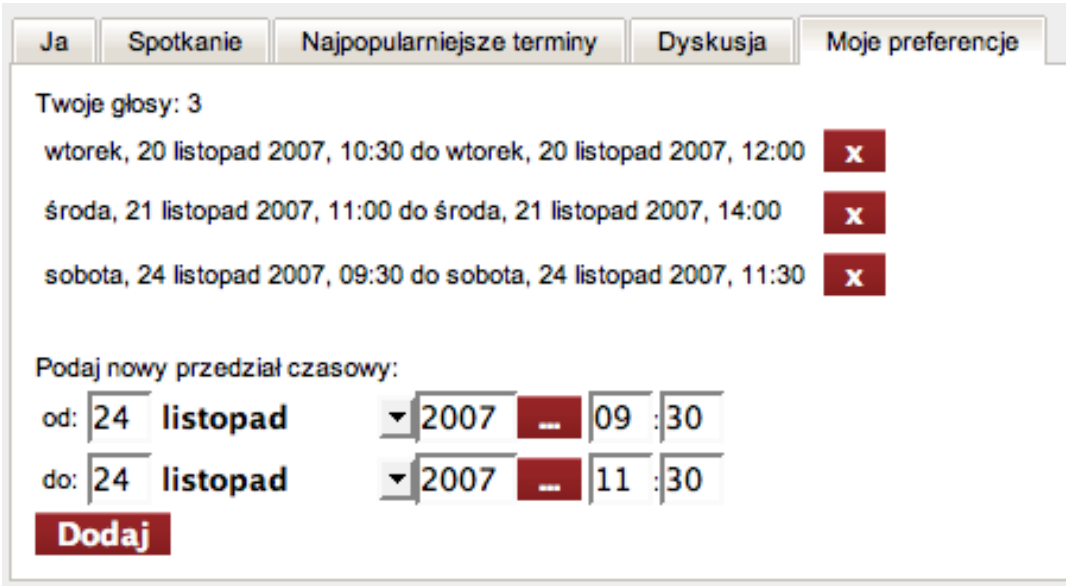


Rysunek 5.6: Zakładka „Najpopularniejsze terminy"

W zakładce „Dyskusja" (rys. 5.7) możliwa jest wymiana wiadomości między uczestnikami. Zakładka „Moje preferencje" (rys. 5.8) prezentuje listę złożonych preferencji oraz pozwala dodawać nowe terminy.

Po zakończeniu składania preferencji uczestnik jest przekierowywany na stronę potwierdzającą przyjęcie preferencji, zawierającą prywatny odnośnik pozwalający na późniejszą zmianę preferencji. Jeśli dana osoba podała wcześniej adres e-mail, to odnośnik ten zostaje jej

Rysunek 5.7: Zakładka „Dyskusja”



Rysunek 5.8: Zakładka „Moje preferencje”

przesłany na podany adres.

5.4. Administracja spotkaniem

Do panelu administracji spotkaniem (rys. 5.10) ma dostęp organizator spotkania dzięki odnośnikowi administracyjnemu, który otrzymał po utworzeniu spotkania. Ekran administracyjny został podzielony na dwie sekcje – „Ogólne" oraz „Akcje". Pierwsza część służy m.in. do zmiany podstawowych parametrów spotkania, które organizator deklarował przy definiowaniu spotkania. Z ważniejszych kwestii można tutaj zmienić długość spotkania, co może mieć wpływ na jego optymalny termin.

Ponadto administrator ma możliwość skreślania uczestników z listy oraz zmiany ich statusu (zwykły lub krytyczny). Zmiany wchodzą w życie dopiero po naciśnięciu przycisku „Uaktualnij", więc istnieje możliwość zrezygnowania z operacji. Usunięcie uczestnika powoduje automatyczne usunięcie wszystkich złożonych przez niego preferencji. Podobnie zmiana statusu danej osoby ma natychmiastowy wpływ na sposób liczenia złożonych przez tą osobę preferencji.

W sekcji zatytułowanej „Akcje" organizator spotkania ma możliwość wykonania dodatkowych operacji: zakończenia głosowania, zmiany listy możliwych terminów oraz skasowanie spotkania.

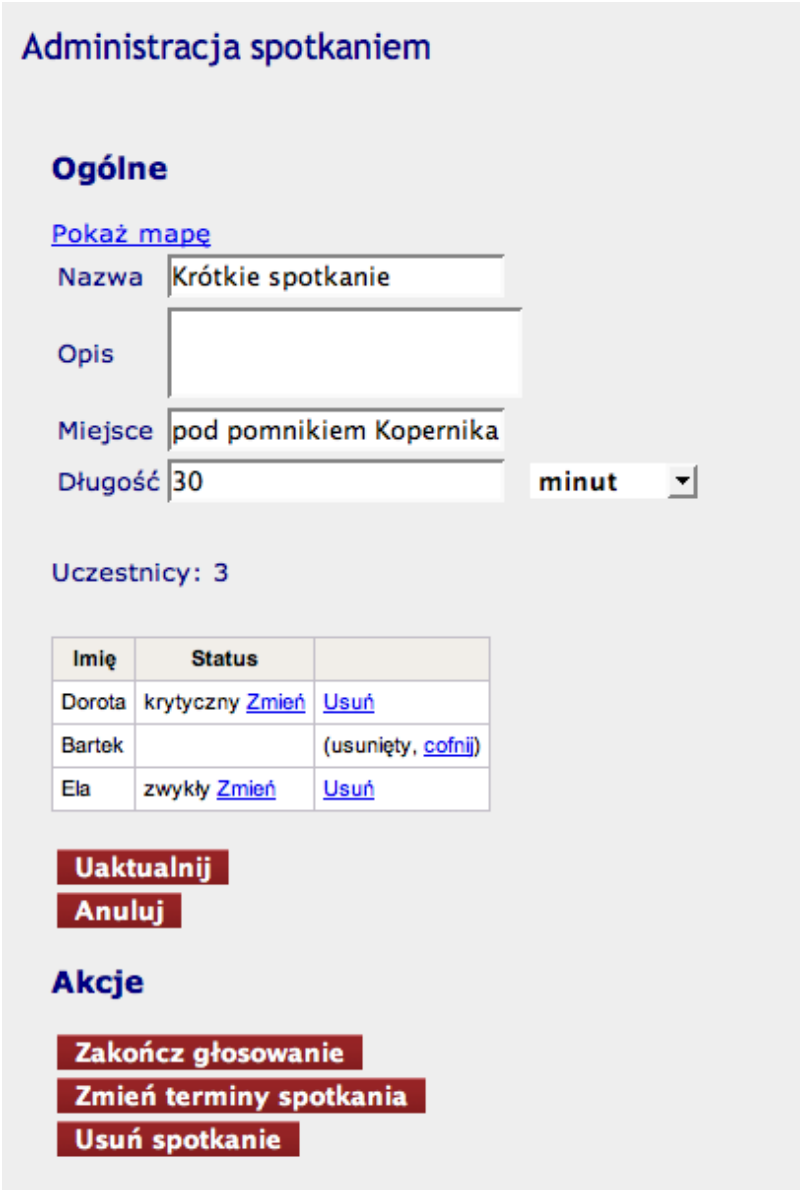
Zmiana listy możliwych terminów przebiega analogicznie do procedury definiowania możliwych terminów opisanej w sekcji 5.2. Usunięcie spotkania wymaga potwierdzenia decyzji przez administratora. Powoduje ono nieodwracalne skasowanie wszystkich danych związanych ze spotkaniem i jego uczestnikami.

Wybranie opcji zakończenia głosowania powoduje przeniesienie użytkownika na stronę, na której jest proszony o potwierdzenie swojej decyzji (rysunek 5.9). System prezentuje mu też najpopularniejszy termin, który po zakończeniu głosowania zostanie uznany za ostateczny termin spotkania. Jeśli istnieje kilka terminów, które są tak samo popularne, jako optymalny zostanie wybrany pierwszy z nich.

Rysunek 5.9: Kończenie głosowania

Organizator spotkania nie musi wybierać zaproponowanego terminu jako daty spotkania. Wybranie opcji „Chcę wybrać inną datę" kieruje administratora do widoku podobnego do widoku wyświetlanego uczestnikowi spotkania (z tą różnicą, że nie można składać preferencji). Administrator może przejrzeć listę najpopularniejszych terminów, ma też do dyspozycji kalendarz z wizualizacją popularności poszczególnych dat. Administrator wybiera ostateczny

termin spotkania i w ten sposób kończy głosowanie. Od momentu zakończenia głosowania dalsze składanie preferencji nie jest możliwe.



Rysunek 5.10: Administracja spotkaniem

Rozdział 6

Przykłady zastosowań

W tym rozdziale opisuję kilka sytuacji z życia codziennego, w których pomocne może być zastosowanie napisanej przeze mnie aplikacji. W zależności od specyfiki spotkania, różne cechy systemu mogą okazać się przydatne przy ustalaniu jego terminu.

6.1. Wybór terminu zajęć dodatkowych

Załóżmy, że grupa studentów musi wspólnie ustalić datę zajęć dodatkowych na uczelni, których termin nie został ujęty w ogólnym planie zajęć. Mogą być to na przykład ćwiczenia, które zwykle odbywają się co tydzień o tej samej porze, ale przepadły z powodu nieobecności prowadzącego i muszą zostać odpracowane w innym terminie.

Wystarczy, że w zaistniałej sytuacji prowadzący wyśle wszystkim studentom w grupie informację o tym, że chciałby, aby zajęcia zostały odpracowane oraz dołączy odnośnik do stworzonej przez siebie strony spotkania. W zdefiniowanym spotkaniu możliwe terminy będą zapewne zależały od tego, w jakich godzina jest dostępna sala, w której mogłyby odbyć się ćwiczenia.

Oczywiście warunkiem koniecznym do obycia się zajęć jest to, aby ich prowadzący mógł się na nich pojawić. Aby zagwarantować, że wybrany termin będzie mu pasował, może on skorzystać z opcji zgłoszenia siebie jako uczestnika krytycznego. Alternatywnie, podczas definiowania spotkania, prowadzący mógłby zawęzić listę terminów, w których może się ono odbyć, do terminów mu pasujących. Zauważmy jednak, iż to drugie rozwiązanie jest nieco mniej elastyczne i nieeleganckie, ponieważ zmiana preferencji prowadzącego pociąga za sobą zmianę możliwych terminów spotkania (których raczej nie należy zmieniać, a szczególnie zawężać w stosunku do początkowych).

W tej sytuacji, uzgodnienie terminu w sposób elektroniczny nie dość, że ułatwi uwzględnienie preferencji wszystkich studentów oraz prowadzącego, to pozwoli na ustalenie terminu dużo szybciej niż na przykład głosowanie dopiero podczas następnych zajęć (na których z kolei części osób może nie być). Szybsze ustalenie terminu może też pozwolić na wcześniejsze odpracowanie ćwiczeń.

6.2. Spotkanie pracowników międzynarodowego zespołu

Wiele korporacji ma swoje oddziały rozproszone po całym świecie. Ustalenie terminu telekonferencji kilku osób pracujących w różnych państwach i na różnych kontynentach może stanowić wyzwanie. Załóżmy, że pracownicy pewnej firmy mieszkają w kilku krajach europejskich oraz w kilku różnych stanach USA.

Z racji różnicy czasu równoczesne spotkanie on-line wszystkich osób może wymagać wynegocjowania wcześniejszego pojawienia się w pracy części Amerykanów lub przedłużenia pracy przez Europejczyków. Nietaktowne byłoby w takiej sytuacji narzucanie terminu z góry, natomiast wysoce pożądane jest ustalenie optymalnego terminu i wzięcie pod uwagę głosu każdej osoby. Zaletą mojej aplikacji może się też okazać możliwość uzgodnienia daty spotkania w sposób w pełni asynchroniczny.

6.3. Ustalenie daty wyjazdu na wakacje

Możliwości zbudowanego przeze mnie systemu nie ograniczają się do uzgadniania terminów krótkich, kilkugodzinnych spotkań. Grupa przyjaciół może w ten sposób wybrać także kilkunastodniowy okres w sezonie wakacyjnym, w którym chce razem spędzić czas.

Uzgodnienie terminu takiego wyjazdu może przebiegać powoli, w międzyczasie możliwości czasowe poszczególnych uczestników mogą się zmieniać. Strona takiego „spotkania" będzie dostępna przez dłuższy czas, oferując każdej zainteresowanej osobie możliwość dokonania korekty preferencji.

Skorzystanie z aplikacji uprości podjęcie decyzji oraz ułatwi wzięcie pod uwagę tego, że na przykład skrócenie wyjazdu o dzień umożliwi niektórym osobom wzięcie w nim udziału. Zgłaszanie preferencji wymusi też podjęcie szybszej decyzji przez poszczególne osoby, więc wcześniej poznamy liczbę uczestników wyjazdu.

6.4. Umówienie się na kawę na czacie

Zdarza się, że chcemy wybrać termin spotkania szybko, właściwie w sposób natychmiastowy, na przykład gdy podczas rozmowy pięciu osób na czacie padnie pomysł pójścia na kawę. Napisana przeze mnie aplikacja jest łatwa i szybka w obsłudze, nie wymaga żadnej rejestracji czy logowania. Stworzenie prostego spotkania nie trwa dłużej niż minutę. Zaraz po tym, korzystając z jednego wspólnego odnośnika, każdy może złożyć swoje preferencje.

Dzięki mechanizmowi uaktualniania wyświetlanych preferencji w tle, po chwili, pozostałe chętne do spotkania się osoby zobaczą, bez potrzeby odświeżania strony, preferencje pozostałych osób, wraz z uaktualnioną wizualizacją najpopularniejszych terminów.

Być może warto będzie zaznaczyć lokalizację kawiarni na mapie umieszczonej na stronie spotkania.

Rozdział 7

Podsumowanie

7.1. Możliwe rozszerzenia

Napisana przeze mnie aplikacja spełnia wszystkie podstawowe wymagania stawiane serwisowi wspomagającemu uzgadnianie terminów spotkań. Rozszerza ona funkcjonalność oferowaną przez aktualnie działające w internecie serwisy tego typu, szczególnie w kwestiach składania dokładnych preferencji oraz ich wizualnej prezentacji. Jednocześnie istnieje wiele obszarów, w których możliwy jest dalszy rozwój aplikacji.

W toku prac udało mi się zaimplementować system wyznaczający najlepsze terminy spotkania uwzględniający tylko jeden rodzaj „wag" preferencji, mianowicie preferencje złożone przez uczestników krytycznych. Dodatkową, ciekawą funkcją byłaby możliwość stopniowania preferencji. Uczestnik miałby możliwość oznaczenia części wybranych przez siebie terminów jako pasujące mu bardziej niż pozostałe lub odwrotnie, wybrane terminy mogłyby być uznawane przez system za pasujące tylko w ostateczności. Osobną kwestią pozostaje tutaj algorytm wyboru optymalnego terminu (można ważyć preferencje lub część z nich brać pod uwagę dopiero przy braku konsensusu) oraz zaprojektowanie interfejsu użytkownika tak, aby dodatkowa funkcjonalność nie wpłynęła ujemnie na łatwość obsługi. Można także rozważyć dodanie opcji umawiania się na spotkania cykliczne (np. cotygodniowe).

Ponadto napisana przeze mnie aplikacja prezentuje wyniki na bieżąco. Być może w niektórych przypadkach przydatnym rozwiązaniem byłaby możliwość zorganizowania głosowania tajnego.

Innym wspomnianym w wymaganiach rozszerzeniem jest współpraca z istniejącymi aplikacjami kalendarzowymi, w szczególności import preferencji z użyciem standardowych formatów lub wysyłanie zaproszeń po zakończeniu głosowania.

Budowa systemu umożliwia łatwe tłumaczenie interfejsu użytkownika na nowe języki, obsługiwany jest standard kodowania UTF-8. W ramach pracy dostarczyłem wersję polską oraz angielską. Nic nie stoi na przeszkodzie, aby wydłużyć listę obsługiwanych języków.

Z kolei oprócz internacjonalizacji interfejsu dobrym pomysłem byłaby możliwość wskazania (lub automatyczny wybór) strefy czasowej, w której mają być wyświetlane daty. Ułatwiłoby to umawianie się na spotkania osobom mieszkającym w oddalonych od siebie miejscach.

Interfejs użytkownika został przetestowany z użyciem przeglądarki internetowej Firefox. Z powodu dużej ilości skryptów napisanych w języku JavaScript (część z nich jest generowana automatycznie przez używane przeze mnie biblioteki), zapewnienie działania aplikacji z dowolną przeglądarką internetową wykracza poza ramy tej pracy. W pierwszej kolejności należałoby dostosować serwis do wymogów najpopularniejszych przeglądarek poza Firefoxem, takich jak Internet Explorer, Opera oraz Safari.

Dołożyłem starań, aby interfejs użytkownika uczynić jak najbardziej przejrzystym i intuicyjnym. W miarę swoich możliwości dobrałem także istniejącą kolorystykę serwisu. Przed umieszczeniem aplikacji w ogólnie dostępnym miejscu w internecie należałoby jednak zadbać o układ i kolorystykę interfejsu zaprojektowaną przez specjalistę w tej dziedzinie.

7.2. Zakończenie

W swojej pracy przeanalizowałem sposoby uzgadniania terminów spotkań oraz istniejące aplikacje, które mogą pomóc w wykonaniu tego zadania. Okazało się, że programom kalendarzowym brakuje wystarczającej funkcjonalności przy wyborze terminu spotkania wielu osób, z których część może korzystać z niekompatybilnych rozwiązań. Z kolei serwisy oferujące grupowe uzgadnianie terminów spotkań działają na zasadzie prostego głosowania, niewystarczającego do dokładnego złożenia preferencji.

Stworzona przeze mnie aplikacja łączy w sobie idee głosowania na terminy z funkcjonalnością kalendarza. Jej główne zalety to możliwość swobodnego deklarowania preferencji czasowych oraz wizualizacja możliwych terminów spotkania z uwzględnieniem ich popularności. Serwis umożliwia późniejszą zmianę złożonych wcześniej preferencji oraz administrację spotkaniem. Dodałem także możliwość zdefiniowania części uczestników jako uczestników krytycznych, czyli takich, którzy bezwzględnie muszą pojawić się na spotkaniu.

Jednym z celów było stworzenie interfejsu użytkownika, który byłby łatwy i intuicyjny w obsłudze. Założenie to zostało osiągnięte dzięki zastosowaniu technologii AJAX. W toku prac doszedłem do wniosku, że istniejące komponenty kalendarza nie oferują wystarczającej funkcjonalności dla mojego projektu, dlatego rozszerzyłem jeden z takich komponentów o możliwość zaznaczania i usuwania przedziałów czasowych bezpośrednio na kalendarzu.

Osobną częścią projektu było stworzenie biblioteki odpowiadającej za operacje na zbiorach interwałów. Elementy tej biblioteki wykonują najważniejszą pracę w całej aplikacji, ponieważ są odpowiedzialne za agregację preferencji złożonych przez poszczególnych użytkowników.

Aplikacja została zbudowana w oparciu o zupełnie nowe rozwiązanie jakim jest szkielet programistyczny Seam. Z kolei użycie sprawdzonej technologii Enterprise JavaBeans 3.0 zapewnia skalowalność serwisu.

Rozważam dalszy rozwój projektu, ukierunkowany na udostępnienie serwisu w internecie. Osiągnięcie tego celu wymaga nie tyle dodania nowej funkcjonalności, ponieważ tą uznaję za wystarczającą, co zatroszczenia się m.in. o szczegóły dotyczące interfejsu i jego wyglądu.

Myślę, że stworzony przeze mnie projekt wpisuje się w ideę tworzenia oprogramowania mającego ułatwiać ludziom życie codzienne. Pozwala on zaoszczędzić czas poświęcony na uzgodnienie terminu spotkania, tym samym pozostawiając więcej czasu na samo spotkanie.

Dodatek A

Zawartość płyty CD

Płyta CD dołączona do niniejszej pracy zawiera następujące pliki i katalogi:

- pwrzeszcz-praca-magisterska.pdf – wersja pracy w formacie PDF,
- jutroniemoge.ear – archiwum binarne z aplikacją (do umieszczenia na serwerze),
- instrukcja.txt – instrukcja instalacji (patrz dodatek B),
- create_db.sql – skrypt tworzący bazę danych, z której będzie korzystać aplikacja,
- src/ – katalog ze źródłami aplikacji (stan na 19 grudnia 2007)
- wymagania/ – katalog zawierający oprogramowanie wymagane do uruchomienia aplikacji:
  - jre-1_5_0_13-linux-i586.bin – Java 1.5 (dla systemu operacyjnego Linux),
  - jboss-4.2.1.GA.tar.gz – serwer aplikacji JBoss,
  - mysql-5.0.45-linux-i686-glibc23.tar.gz – baza danych MySQL (dla systemu operacyjnego Linux),
  - mysql-connector-java-5.0.5-bin.jar – sterownik bazy danych MySQL.

Dodatek B

Instalacja

Aplikację można uruchomić pod dowolnym systemem operacyjnym, dla którego istnieje implementacja Javy oraz dystrybucja bazy danych MySQL. W szczególności może to być Microsoft Windows, Linux czy Mac OS X.

W niniejszym dodatku została opisana procedura instalacji dla Linuksa. Dla pozostałych systemów należy ściągnąć z internetu (http://www.java.com/download, http://dev.mysql.com/downloads) i zainstalować odpowiednie wersje wymienionego oprogramowania.

Instalacja przebiega w następujących krokach. Należy:

- Sprawdzić posiadaną na komputerze wersję Javy. Wymagana jest wersja 1.5. W razie jej braku należy zaistalować Javę, wykonując plik wymagania/jre-1_5_0_13-linux-i586.bin i postępując zgodnie z instrukcjami.
- Zainstalować bazę danych MySQL w wersji 5.0. Archiwum zawierające pakiet z bazą danych zostało umieszczone na płycie CD w pliku mysql-5.0.45-linux-i686-glibc23.tar.gz.
- Utworzyć nową bazę danych i nowego użytkownika bazy oraz przypisać mu uprawnienia do tej bazy. W tym celu wystarczy wykonać polecenie „mysql" (dla użytkownika z odpowiednimi uprawnieniami) oraz wykonać skrypt „create_db.sql".
- Jeśli chcemy wybrać inne niż domyślne wartości dla nazwy bazy danych oraz nazwy i hasła użytkownika, to należy uprzednio wpisać je w odpowiednie miejsca w skrypcie oraz zmienić wpisy w pliku jutroniemoge-prod-ds.xml w źródłach aplikacji.
- Rozpakować w wybranym przez siebie miejscu archiwum z serwerem aplikacji JBoss (wymagania/jboss-4.2.1.GA.tar.gz). W dalszej części zakładamy, iż jest to katalog jboss.
- Z katalogu wymagania skopiować plik mysql-connector-java-5.0.5-bin.jar do katalogu jboss/server/default/lib.
- Skopiować plik jutroniemoge.ear do katalogu jboss/server/default/deploy.
- Skopiować plik src/jutroniemoge/resources/jutroniemoge-prod-ds.xml do katalogu jboss/server/default/deploy.
- Uruchomić serwer aplikacji JBoss (komenda jboss/bin/run.sh).
- Poczekać, aż serwer zgłosi gotowość do pracy.

Po wpisaniu w przeglądarce internetowej adresu http://localhost:8080/jutroniemoge uzyskamy dostęp do aplikacji.

Bibliografia

[1] 30 Boxes, http://www.30boxes.com

[2] Apache Ant, http://ant.apache.org

[3] Apache Tomcat, http://tomcat.apache.org

[4] Bedework: open source calendar for the enterprise, http://www.bedework.org

[5] Bergsten H., *Improving JSF by Dumping JSP*, http://www.onjava.com/pub/a/onjava/2004/06/09/jsf.html, ONJava 2004

[6] Burke B.,*EJB 3.1: No-interface view feedback*, http://bill.burkecentral.com/2007/09/24/ejb-31-no-interface-view-feedback

[7] Burke B., Monson-Haefel R., *Enterprise JavaBeans 3.0, Fifth Edition*, O'Reilly 2006

[8] Burns E., Kitain R. *JavaServer Faces Specification, Version 1.2*, Sun Microsystems, 2006

[9] Daboo C., Desruisseaux B., Dusseault L., *Calendaring Extensions to WebDAV (CalDAV)*, Request for Comments (RFC) 4791, http://tools.ietf.org/html/rfc4791, 2007

[10] Dawson F., Stenerson D., *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*, Request for Comments (RFC) 2445, http://tools.ietf.org/html/rfc2445, 1998

[11] DeMichel L., Keith M., EJB 3.0 Expert Group *JSR 220: Enterprise JavaBeans, Version 3.0*, Sun Microsystems, 2005

[12] Doodle, http://www.doodle.ch

[13] Garret J., *Ajax: A New Approach to Web Applications*, http://www.adaptivepath.com/ideas/essays/archives/000385.php, 2005

[14] Google Calendar, http://www.google.com/calendar

[15] Google Maps API, http://www.google.com/apis/maps

[16] Hibernate, http://www.hibernate.org

[17] HSQLDB, http://www.hsqldb.org

[18] iCal, http://docs.info.apple.com/article.html?artnum=304766

[19] ICEFaces, http://www.icefaces.org

[20] Java Platform, Enterprise Edition, Sun Microsystems, http://java.sun.com/javaee/index.jsp

[21] JavaServer Facelets, https://facelets.dev.java.net

[22] JBoss Application Server, http://labs.jboss.org/jbossas

[23] JBoss RichFaces, http://www.richfaces.org

[24] JBoss Seam, http://labs.jboss.org/jbossseam

[25] JScience, http://www.jscience.org

[26] JScience Advanced Developement Experimentation, https://jade.dev.java.net

[27] JUnit, http://www.junit.org

[28] Lotus Notes, http://www.lotus.com/notes

[29] Meet-O-Matic, http://www.meetomatic.com

[30] Meeting Wizard, http://meetingwizard.com

[31] Microsoft Office Outlook 2007, http://office.microsoft.com/pl-pl/outlook/FX100487751045.aspx

[32] Monson-Haefel R., *BZ Research on Java EE Market Share Data*, http://rmh.blogs.com/weblog/2006/05/bz_research_on_.html

[33] Mozilla Firefox, http://www.mozilla-europe.org/pl/products/firefox

[34] Mozilla Lightning, http://www.mozilla.org/projects/calendar/lightning

[35] Mozilla Sunbird, http://www.mozilla.org/projects/calendar/sunbird

[36] MyFaces Tomahawk, Apache Software Foundation, http://myfaces.apache.org/tomahawk

[37] MySQL, http://www.mysql.com

[38] Simile Timeline, http://simile.mit.edu/timeline

[39] TestNG, http://www.testng.org

[40] Thurrott P., *Windows Vista Beta 2 Review, Part 3. New Features*, http://www.winsupersite.com/reviews/winvista_beta2_03.asp

[41] Web Beans Expert Group, *JSR 299: Web Beans, Early Draft Review*, 2007

[42] Web Calendar Access Protocol Overview, http://docs.sun.com/source/817-5698/pmWCAPov.html, Sun Microsystems, 2004

[43] Windows Calendar, http://www.microsoft.com/windows/products/windowsvista/features/details/calendar.mspx

[44] Yahoo! Calendar, http://calendar.yahoo.com

[45] Yuan M., Heute T., *JBoss Seam, Simplicity and Power Beyond Java EE*, Prentice Hall 2007