SpringAtom – Reference Document

Koncept

Aplikacja **SpringAtom**[1] jest rozszerzeniem aplikacji **AgatomProject**[2]**,** której głównym zamysłem było dostarczenie programu wspierającego zarządzania terminarzem wizyt w warsztacie samochodywym. Z tego powodu aplikacja posiadała następującą fukcjonalność:

- Skoncentrowanie się na obiekcie samochodu, jako głównym elemencie logiki biznesowej – *samochód posiada właściciela, nie na odwrót*
- Zarządzania bazą wizyt
- Zarządzania bazą klientów firmy, dla których umawiano wizyty
- Wsparcie dla historii wizyt konkretnego klienta lub konkretnego samochodu
- Kontrolę wiarygodności klientów

Całość została zaimplementowana w języku C++ przy wsparciu biblioteki Qt (zarówno logika oraz widok), a do warstwy danych posłużyła relacyjna baza danych MySQL.

Mimo potocznego faktu – program działa – istnieje w nim wiele aspektów, które zostały oprogramowane w sposób niepoprawny, bądź mało wydajny. Przytoczyć można chociażby architekturę aplikacji – **desktop** – co skutecznie uniemożliwia jej użycie na skalę szerzą niż wewnątrz firmy. To, a także wiele innych problemów, wynikających z konstrukcji oraz implementacji poszczególnych modułów, przyczyniło się bezpośrednio do idei **SpringAtom**.

SpringAtom

SpringAtom ma być aplikacją dającą tę samą funkcjonalność, jaką oferował **AgatomProject**, ale zaserwowaną w odsłonie, umożliwiającą użytkowania programu jako aplikacji webowej.

Funkcjonalność

- Samochód elementem nadrzędnym
    - Wersjonowanie obiektów typu samochód, co daje możliwość śledzenia historii własności danej jednostki oraz pozwala na tworzenie wizyt przypisanych do konkretnej rewizji, a tym samym do klienta, który w danym momencie był właścicielem
    - Tworzenie spotkań przypisanych do samochodu, a nie do klienta
- Spotkanie jako obiekt posiadający wiele zadań
    - Zadania opisują co zostało wykonane przy konkretnej wizycie
    - Spotkania przypisane jest do konkretnej rewizji samochodu
    - Spotkania powiązane jest z dwoma mechanikami, niekoniecznie tymi samymi
        - Pierwszy wskazuja na osobę, która umówiła dane spotkania
        - Drugi wskazuję na osobę, które została oddelegowana, przyjęła dane zlecenie do realizacji
- Klienci przestają być biernym elementem systemu
    - Mogą podglądać historię swoich wizyt
    - Mogą wystosowywać żądanie nowej wizyty
- Wewnętrzny system notyfikacji

---

[1] SpringAtom
[2] AgatomProject

    - Notyfikacje wysyłane do klientów i/lub mechaników zależnie od potrzeby. Dany użytkownik po zalogowaniu otrzymywałby komunikat o nieodczytanych notyfikacjach

Technologia

- Spring Framework
- Spring Web Flow
- Spring MVC
- GWT lub GXT (ExtJS for GWT)
- MySQL
- Hibernate
- Spring Security
- Spring Batch [przetwarzanie większych porcji danych]
- LDAP (Spring LDAP) [możliwe] z uwagi na ACL

Założenia

- Wykorzystania AJAX
- Modularna budowa widoku w oparciu o zawierane JSP
- Widok renderowany zależnie od uprawnień użytkowników (ukrywania, blokowanie akcji bazując na rolach)
- Kontrola dostępu
- Możliwie najwięcej akcji dostępnych do wykonania poprzez Wizardy
- Możliwość rejestracji nowych klientów (bardziej w kontekcie bycia użytkownikiem systemu, niż nowym klientem)
- Bezpieczeństwo danych [możliwie najwięcej danych zabezpieczonych hashowaniem]
- Bezpieczny protokół HTTPS
- Ochrona przed atakami na aplikacje WEB [np. SQLInjection]
- Migracja danych ze starego systemu
- Nacisk na konfigurowalność aplikacji [edycja plików properties, a nie plików *.xml Spring'a]
- Uproszone narzędzia administracyjne dostępne z poziomu konsoli
    - Ładowania danych typu **metaData**
    - Kontrolowania uprawnień użytkowników
    - Zarządzania zmiennymi z plików properties

Założenia rozwojowe

- Integracja kont klienckich z mechanizmem OAuth [możliwe logowanie do systemu dzięki np. Kontu pocztowemu GMail]
- Integracja systemu [SpringIntegration] z aplikacją mobilną na system Android [dającą dostęp do podstawowych funkcji dla klientów]
- Kontrola wizyt klientów, podpowiadania kolejnych wizyt na podstawie informacji o przebiegu samochodu itp

Problemy

Najwięcej problemów będzie najpewniej z technologiami i modułami Springa, których do tej pory nie miałem przyjemności używać w stopniu zaawansowanym. Dobrym przykładem, niezwiązanym być może w 100% ze Spring, jest wersjonowanie konkretnych obiektów. Jest to funkcjonalność rodzaju **must-have**, dlatego zamierzam ją zaimplementować w sposób możliwie najbardziej generyczny i zautomatyzowany. Kolejnym problemem, który przewiduje będzie integracja bibliotek **client-side**, takiej jak chociażby GXT na potrzeby stworzenia widoków, zbliżonych do tych znanych z aplikacji desktopowych. Ostatecznie, na obecną chwilę, największym problemem – blockerem – będzie problem

migracji danych ze starego systemu do nowego. Mimo pewnych podobieństw, schemy obu baz danych różnią się, ponieważ w nowej starałem się położyć jak największy nacisk na obniżenie redundancji danych, czego nie udało mi się osiągnąć w wersji pierwotnej.