On-line jízdní řády: závěrečná zpráva

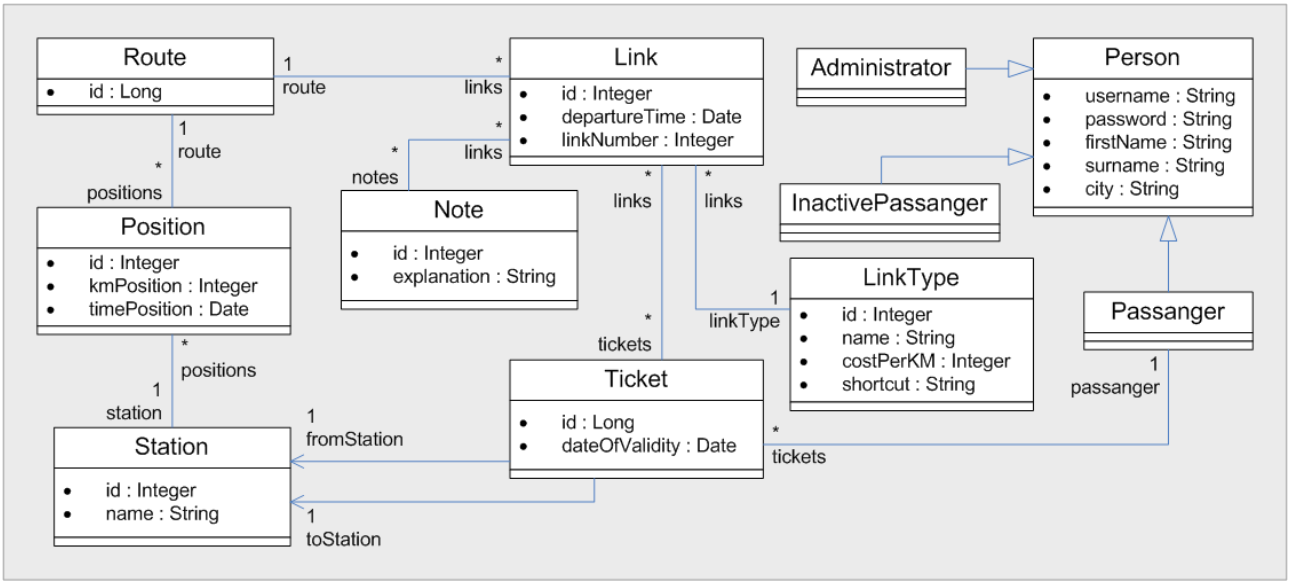
Jan Fabián & Martin Lukeš
X33EJA zimní semestr 2010/2011

V průběhu zpracovávání projektu jsme narazili na některé zajímavosti – prolémy týkající se EE Javy, databáze a teamové spolupráce.

Při práci v našem malém teamu jsme valnou část projektu zpracovali podle zásady agilní metodiky vývoje Pair programming - viz http://en.wikipedia.org/wiki/Pair_programming. Tato metodika nám vyhovovala a byla celkově přínosná.

Poznámka: insert skript s testovacími daty (*insert.sql*) se nachází v kořenovém adresáři projektu (deploy jej nespouští automaticky).

Databáze jízdních řádů – úpravy během vývoje



Obrázek 1: struktura databáze

Základní struktura databáze se od návrhu neměnila. Nejzásadnější změnu si vyžádala správa uživatelských rolí, kdy byla přidána tabulka Person, od které dědí jednotlivé skupiny uživatelů (Administrator, Passanger a InactivePassanger).

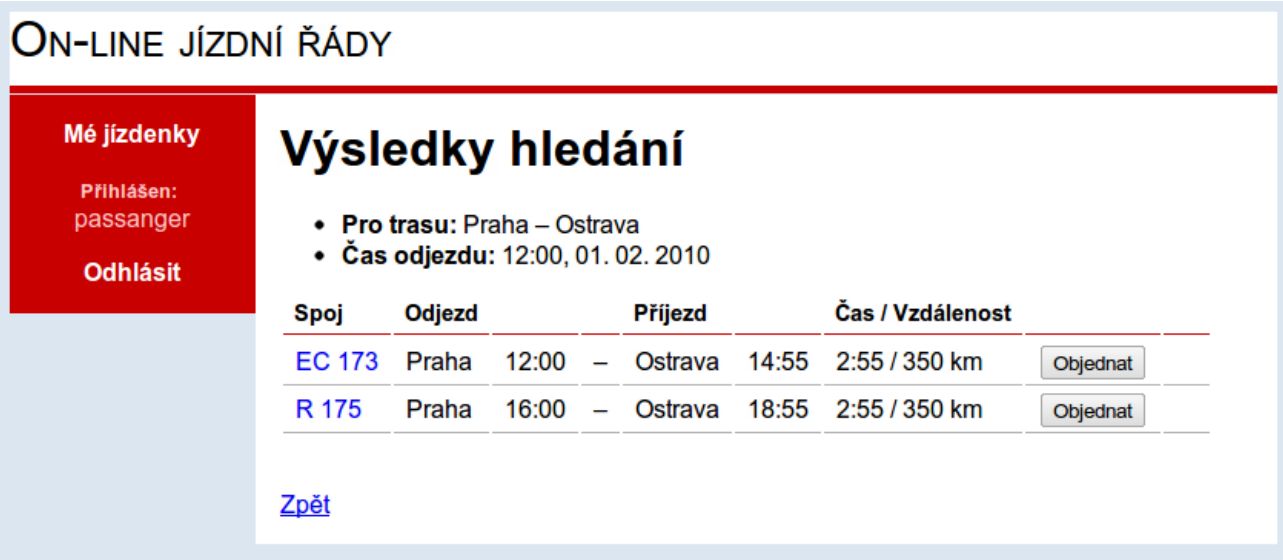
Během vývoje se jako značně problematický jevil návrh relací mezi jízdenkou (*Ticket*) a ostatními tabulkami. Existuje totiž relace M:N mezi *Ticket* a *Link*, tedy jízdenka může být přestupní, ovšem každá jízdenka má určenu pouze výchozí (*fromStation*) a cílovou (*toStation*) stanici. Nejsou tedy určeny přestupní body a trasa projetá v rámci jízdenky tak může být nejednoznačná. Protože ke zjištění tohoto problému došlo již v poměrně pokročilé fázi vývoje, nebylo již zasahováno do databáze a místo toho byl tento problém ošetřen na aplikační úrovni, kdy se přestupní místa zjistí automaticky (viz metoda *TimetableSession.get*{*arriving*|*departing*}*stationByLink*):

- potkávají-li se trasy dvou v rámci jízdenky po sobě následujících spojů pouze v jednom bodě, je tento přestupní stanicí
- potkávají-li se tyto trasy ve více bodech, je jako přestupní určen první

- k situaci, že se tyto trasy nepotkávají v žádném bodě, by nemělo dojít, neboť aplikace nedovolí pro jízdenku takové spoje v daném pořadí vybrat.

Ani jeden z nás se předtím nezabýval detailněji problémem času, data v Javě. V naší aplikaci se čas používá ve velké míře. Různé datové typy používané pro časové údaje v Javě v kombinaci s omezeními danými jejich použitím v persistent entitách pro nás byly drobným problémem, který se nicméně podařilo překonat. Neznamená to, že řešení práce s daty a časem je ideální a jako autoři aplikace jsme si vědomi, že neřešíme problém komplexně, jen v rámci vyhledávání spoje v jeden den. Detailnější vyhledávání pro různá data a časové pásmo zanedbáváme.

Implementace



Obrázek 2: ukázka aplikace

EJB modul obsahuje dvě SessionBeans, z nichž jedna je stateful a druhá stateless. Stateful bean *TicketSession* poskytuje business metody pro práci s jízdenkami (objednávání i prohlížení), stateless bean *TimetableCenter* obsahuje business logiku pro zbylé části aplikace (zejména vyhledávání spojů, ale i administrace, registrace cestujících apod.).

Postatnými BackingBeans jsou zejména *SearchingBean*, *OrderBean* a dále *ListBean* a *PassangerBean*. Zbylé BackingBeans jsou využívány jen stránkami v administrátorské sekci Správa jízdního řádu (viz následující poznámka o implementaci administrátorského rozhraní).

Proti plánu bylo částečně bylo implementováno i administrátorské rozhraní, především však v rámci testů na počátku vývoje, a dále již nebylo vyvíjeno a laděno. S výjimkou správy uživatelů (která je plně funkční, viz dále) tak není příliš použitelné; u definitivního administrátorského rozhraní by mj. bylo vhodné oprostit se pouze od přesného zobrazování tabulek v databázi a přejít k editaci logických celků (např. *Route* a příslušné *Positions* jako celek).

Vyhledávání

Základem vyhledávání jsou metody *TimetableCenter.getLinksForJourney* , *TimetableCenter.getRoutesConnectingStations* a *TimetableCenter.getLinksByDepartureTime.* Vyhledávají se pouze spoje bez přestupu. Nejprve jsou nalezeny všechny trasy, které z výchozí do

cílové stanice vedou, pro každou z těchto tras jsou pak nalezeny spoje, které z výchozí stanice odjíždějí ne dříve než ve vybraný čas.

Správa jízdenek

Problémem, způsobeným jednak způsobem uložení jízdenek v databázi (viz Databáze – Úpravy během vývoje), jednak způsobem vyhledávání přestupních spojů, je nemožnost objednání libovolné jízdenky. Uživatel může například objednat jízdenku nejprve na spoj Praha – Přerov – Ostrava a dále přidat přestupní spoj Ostrava – Přerov – Brno. Do jízdenky jsou však uloženy jen koncové stanice Praha a Brno. Jako přestupní stanice bude místo Ostravy určen Přerov (první stanice, kde je možný přestup mezi oběma spoji).

Tento problém je řešen již při objednávání jízdenky, kdy je uživateli zobrazena již upravená varianta trasování (v tomto případě přes Přerov) a dále aplikace jízdenku zobrazuje vždy s přestupem v Přerově. Toto chování není příliš na závadu, neboť vlastně dochází jen k určité optimalizaci nevhodných tras, které byly nalezeny kvůli ručnímu hledání přestupních spojů. Přesto se jedná o spíše provizorní řešení, které bylo ovšem vynuceno zjednodušeným vyhledáváním přestupních spojů. Viz metoda *TimetableSession.addLinkToTicket*.

Správa uživatelů

Nepřihlášený uživatel se pomocí příslušného formuláře může do systému registrovat. Po registraci je uživatel uložen do databáze jako *InactivePassanger* a k účtu se není možné přihlásit. Je nejprve vyžadována aktivace uživatele. Ta může být provedena administrátorem v sekci Správa uživatelů, kdy je původní záznam *InactivePassanger* smazán a je vytvořen nový objekt *Passanger* se stejnými atributy.

Docházelo zde k problému, způsobeného patrně faktem, že původní mazaný *InactivePassanger* i nový *Passanger* mají stejná usernames, což jsou primární klíče, a nový *Passanger* nebyl do databáze přidán. Problém byl vyřešen přidáním prefixu *inactive_* před username uživatele *InactivePassanger* při registraci, který se při aktivaci „odmaže“. Ověřování jedinečnosti username při registraci se tak ale nemůže dít na úrovni databáze přes primární klíče a k ověření tedy dochází v těle metody *TimetableCenter.updateInactive.*