Mendelova univerzita v Brně
Provozně ekonomická fakulta

Softwarové inženýrství

Kuchařka

Bc. Roman Koláček
Bc. Václav Steiger
Bc. Michal Vachler

Brno 2015

Obsah

Obrázky

1 Specifikace

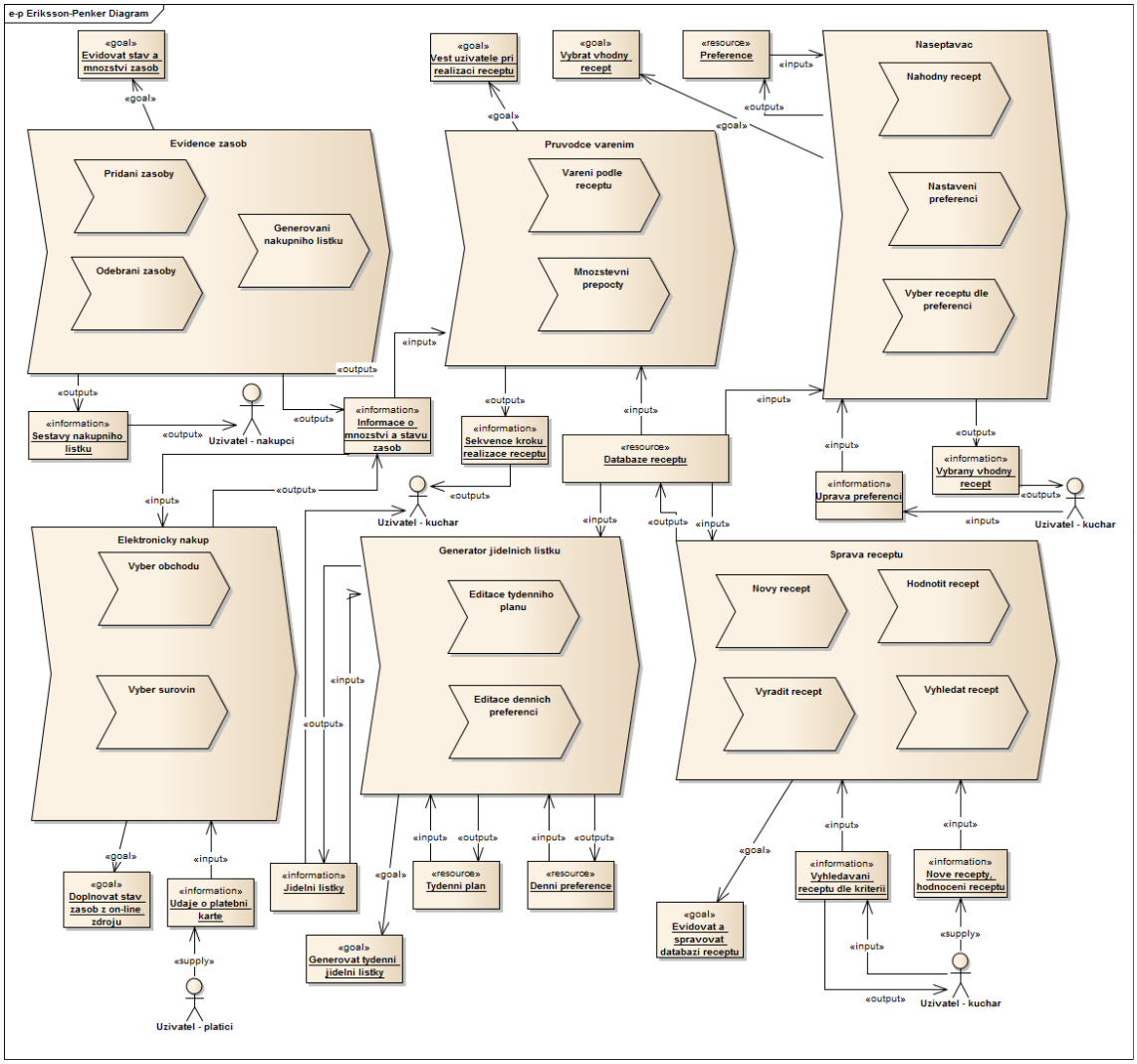
1.1 Funkční požadavky

- Správa receptů – systém vede databázi receptů na pokrmy, umožňuje jejich vkládání, mazání, úpravu.
- Správa zásob – systém vede databázi skladových zásob potravin, umožňuje jejich vkládání, vyřazování, automaticky odepisuje pokrmy, ze kterých byl uvařen recept.
- Elektronický nákup – systém umožňuje elektronicky nakupovat chybějící zásoby s důrazem na co nejvyšší míru autonomie
- Našeptávač – systém je schopen navrhovat recepty k použití.

1.2 Nefunkční požadavky:

- Autonomie – důraz kladen na co nejvyšší autonomii systému, systém má ulehčit, ne komplikovat činnosti, které se vážou k funkčním požadavkům
  - Systém pomocí Našeptávače automaticky radí recepty
  - Elektronický nákup automaticky objednává zásoby, v celém procesu vyžaduje minimální asistenci uživatele
- Udržitelnost a snadná rozšiřitelnost – je třeba systém navrhnout takovým způsobem, aby byl v budoucnu snadno rozšiřitelný o další funkce
  - Modularita – systém rozdělen do jednotlivých modulů, lze přidávat další moduly do již funkčního celku
- Ergonomie a intuitivní používání – je třeba systém navrhnout tak, aby byl jednoduše a intuitivně použitelný i pro méně počítačově gramotné uživatele
  - Logická a designová jednotnost - systém navrhován tak, aby logická stavba úkonů při jeho používání byla co nejpřímější a nejjednodušší.

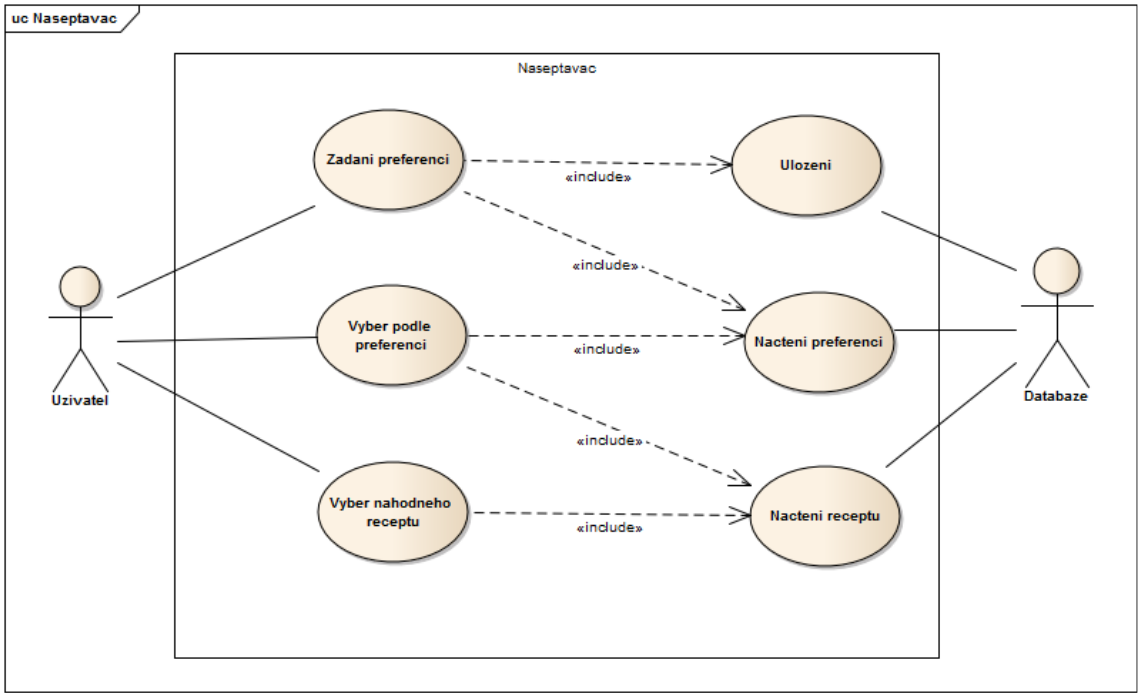
2 Eriksson-Penker diagram



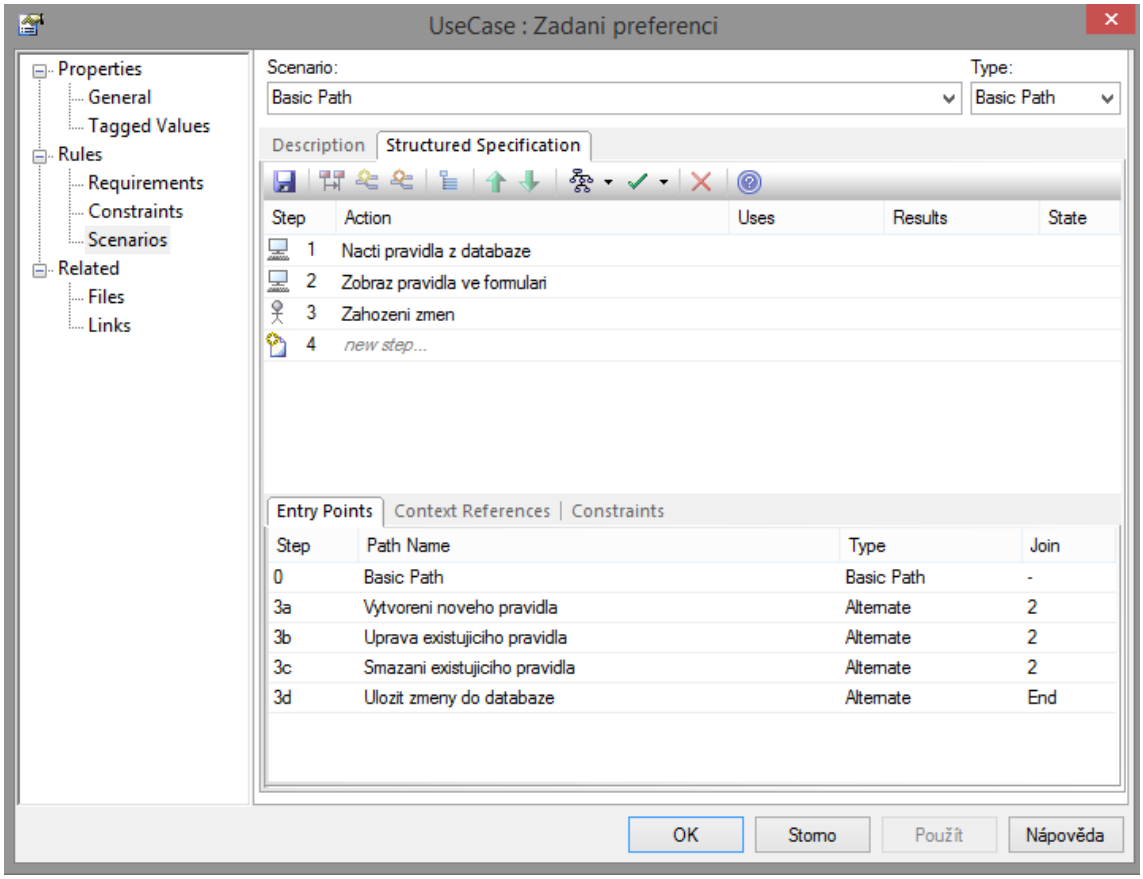
Obrázek 1 – Eriksson-Penker diagram

1 Use Case

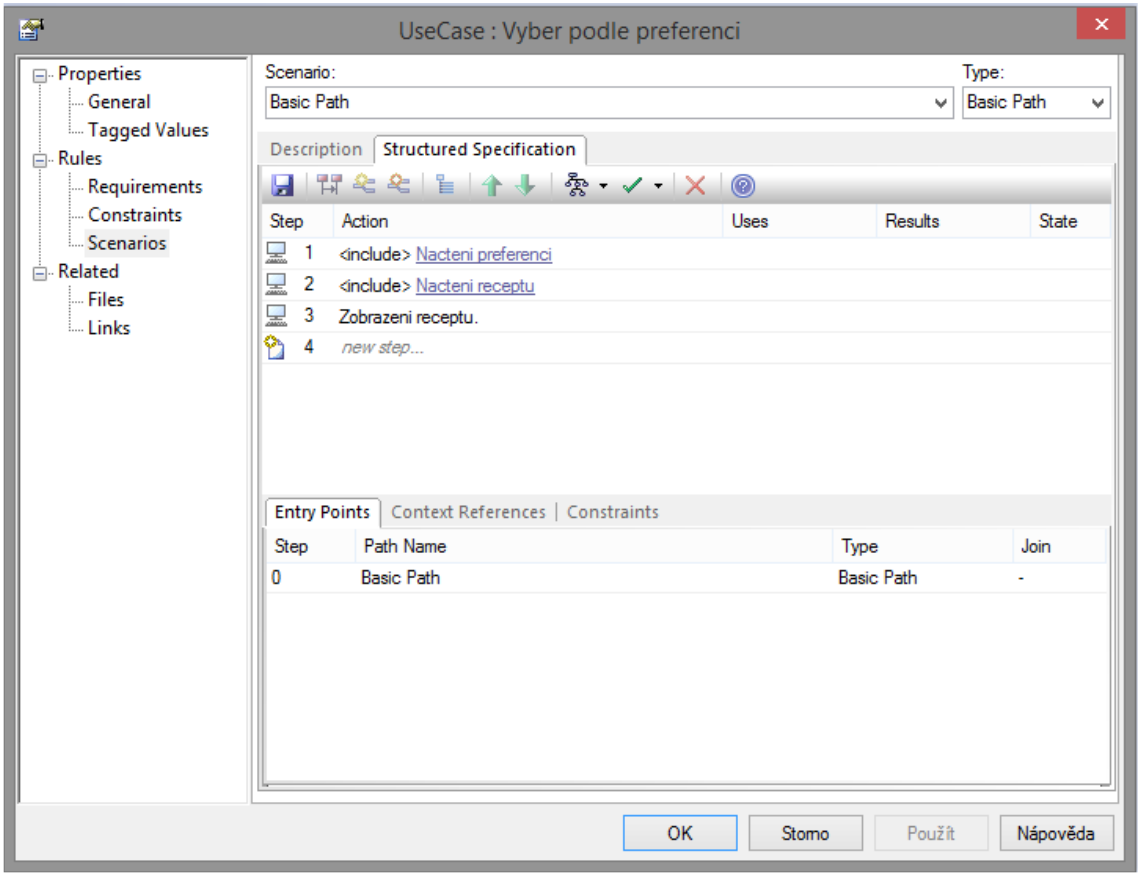
Use case diagram znázorňuje interakci mezi aktéry a systémem – v našem případě jsou aktéry Uživatel a Databáze. V tomto diagramu popisujeme fungování našeptávače, kdy Uživatel vytváří a nebo vybírá preference, aby mu systém mohl nabídnout relevantní recept.



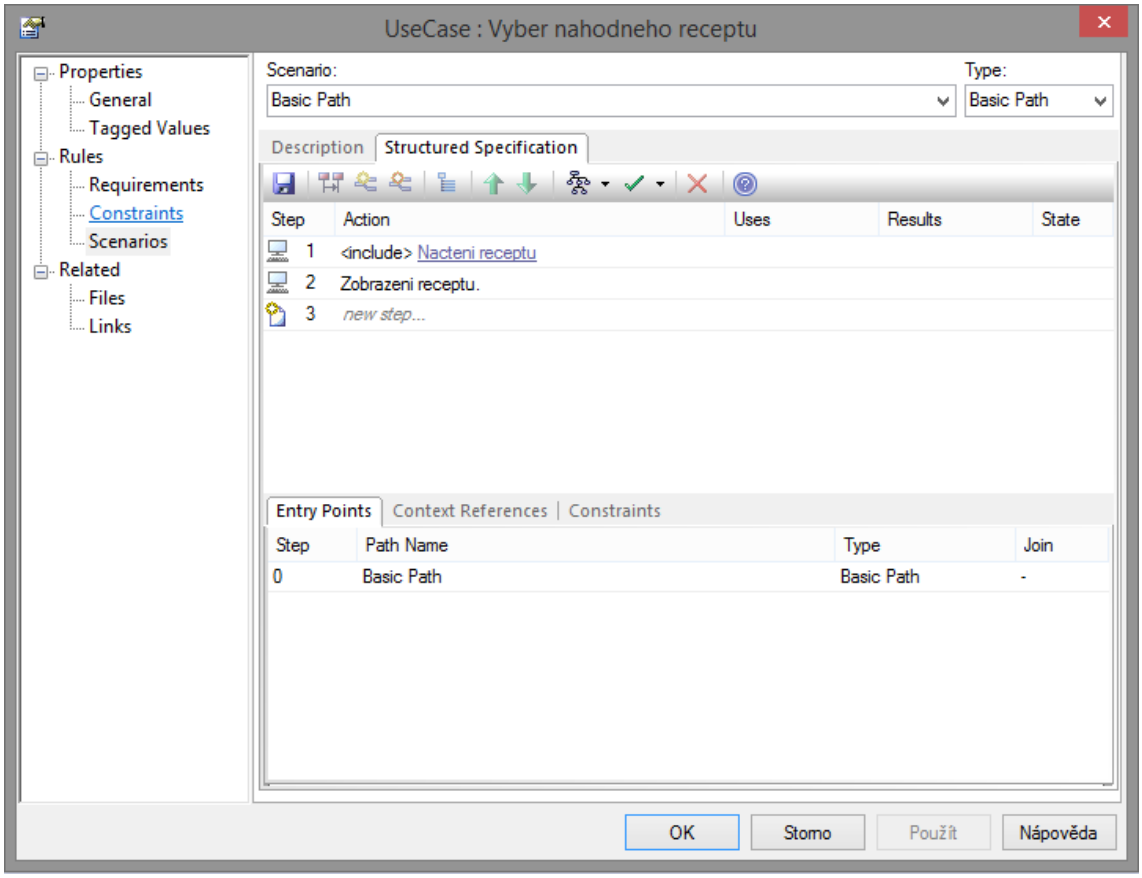
Obrázek 2 – Use Case: Našeptávač



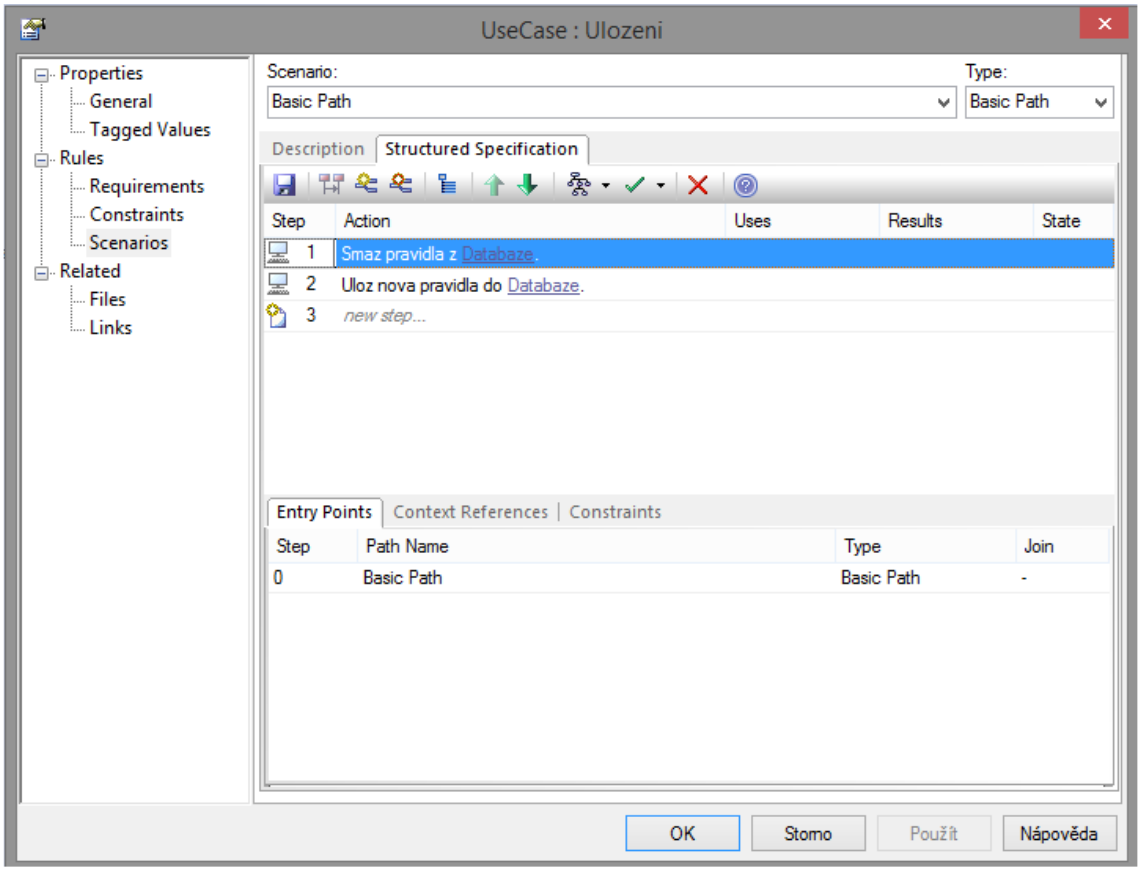
Obrázek 3 – Scénář: Zadání preferencí



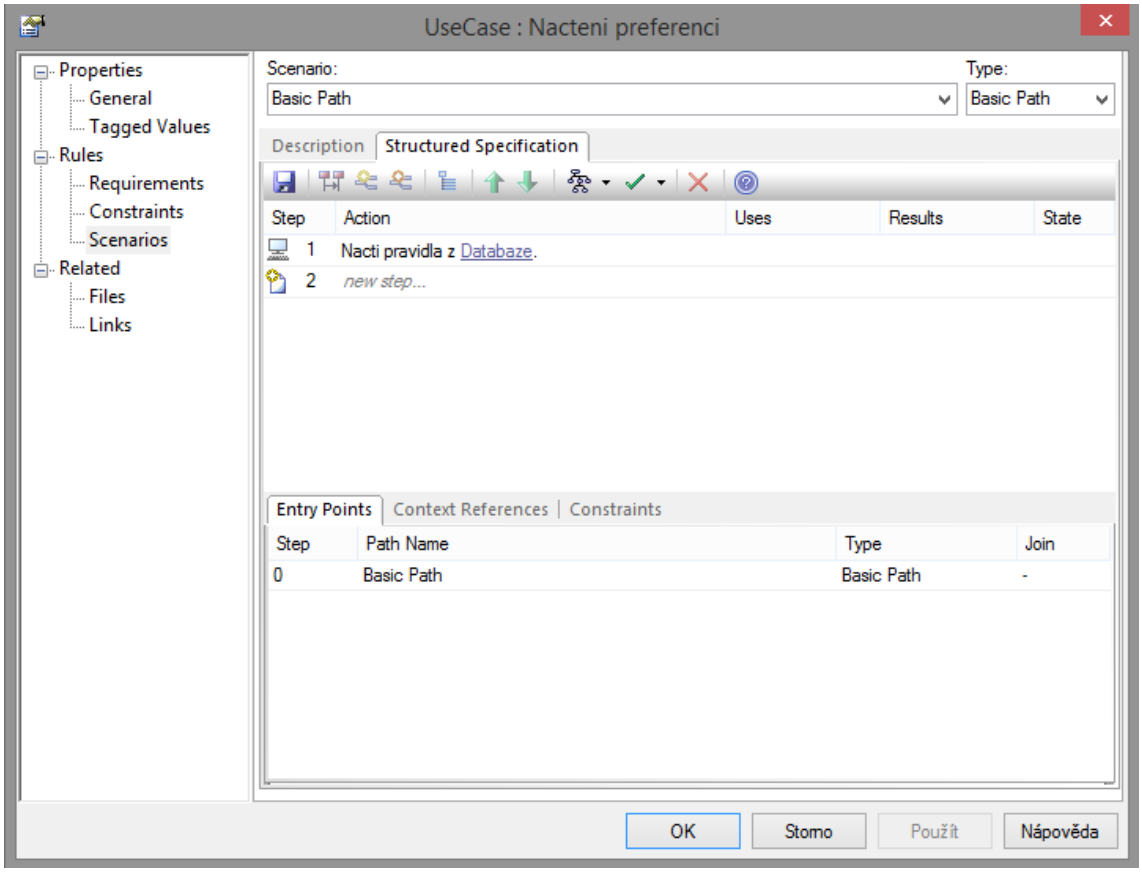
Obrázek 4 – Scénář: Výběr podle preferencí



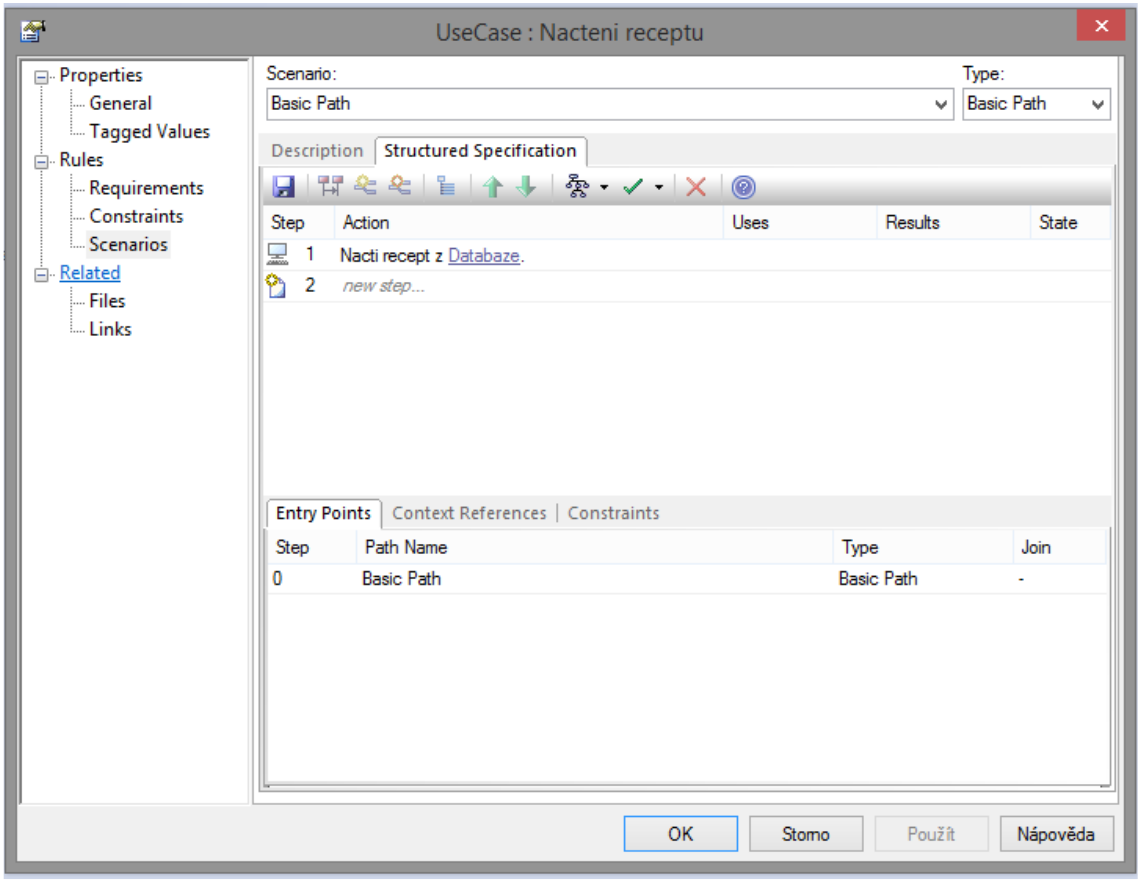
Obrázek 5 – Scénář: Výběr náhodného receptu



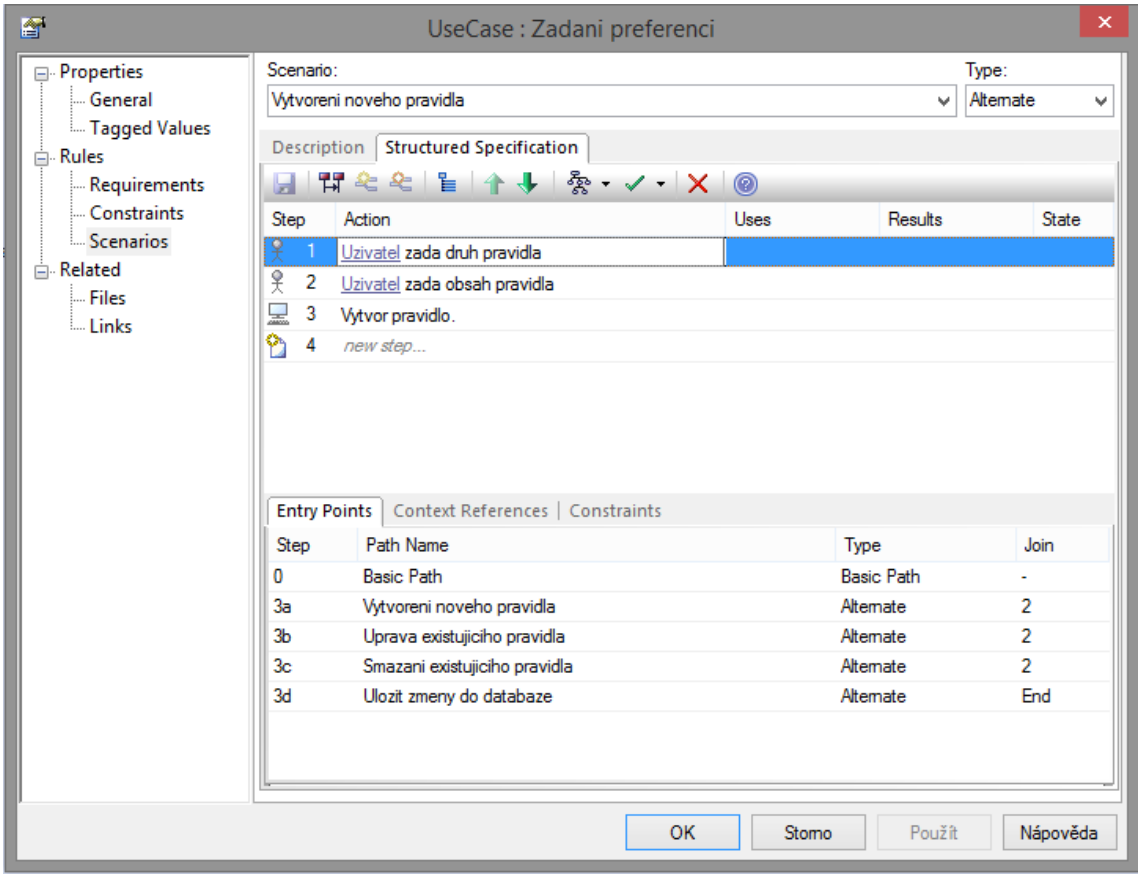
Obrázek 6 – Scénář: Uložení



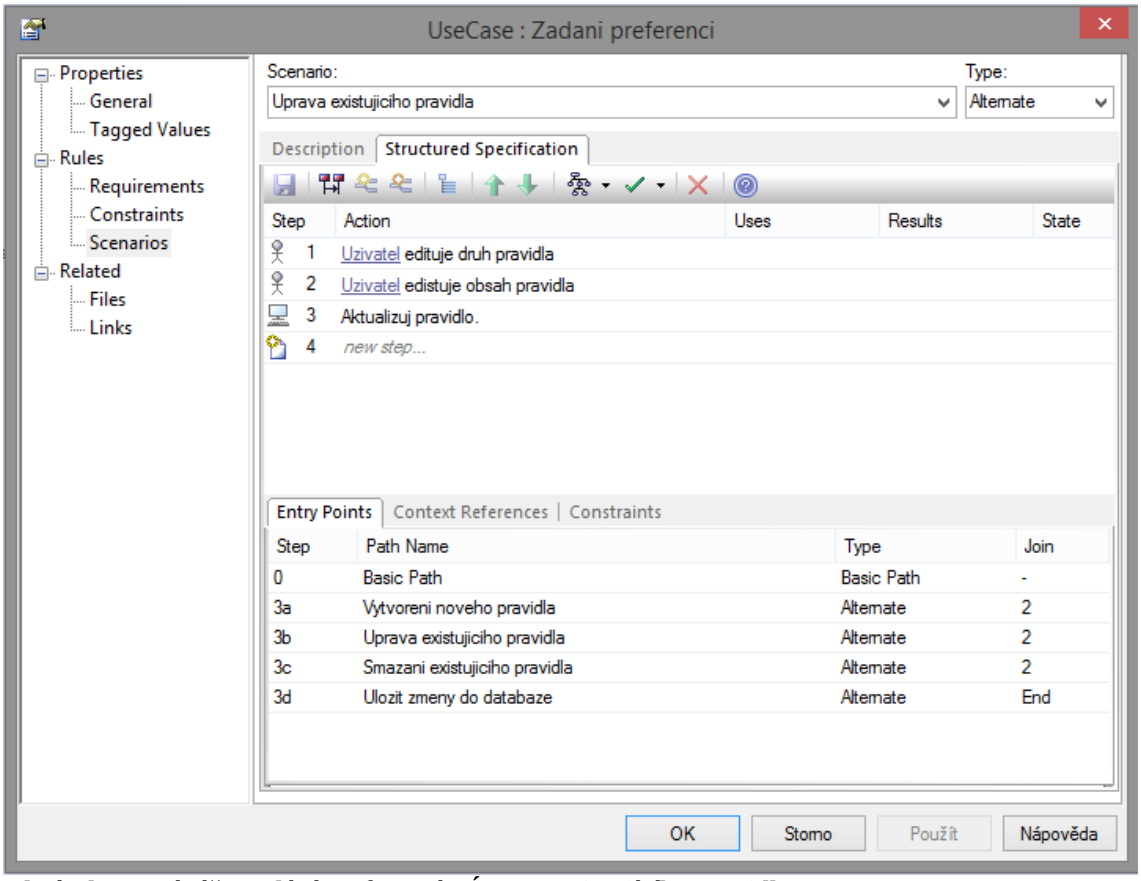
Obrázek 7 – Scénář: Načtení preferencí



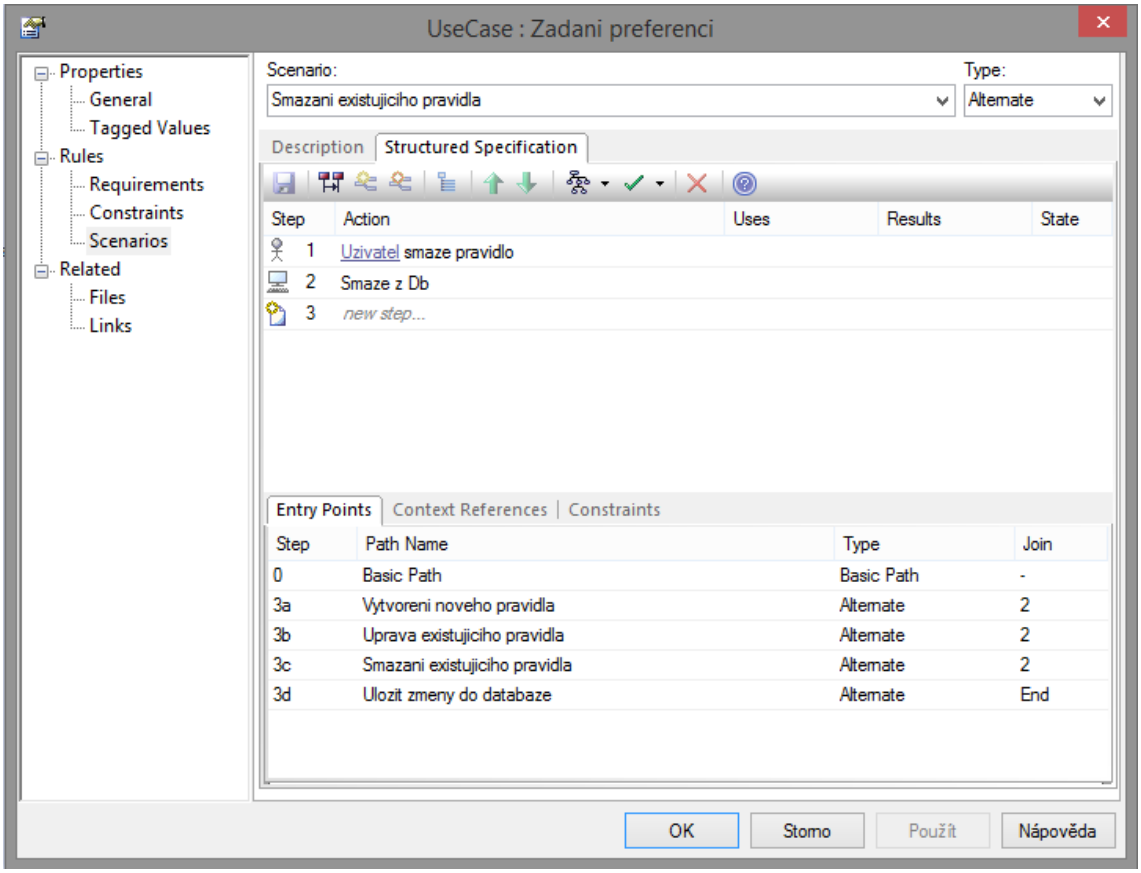
Obrázek 8 – Scénář: Načtení receptu



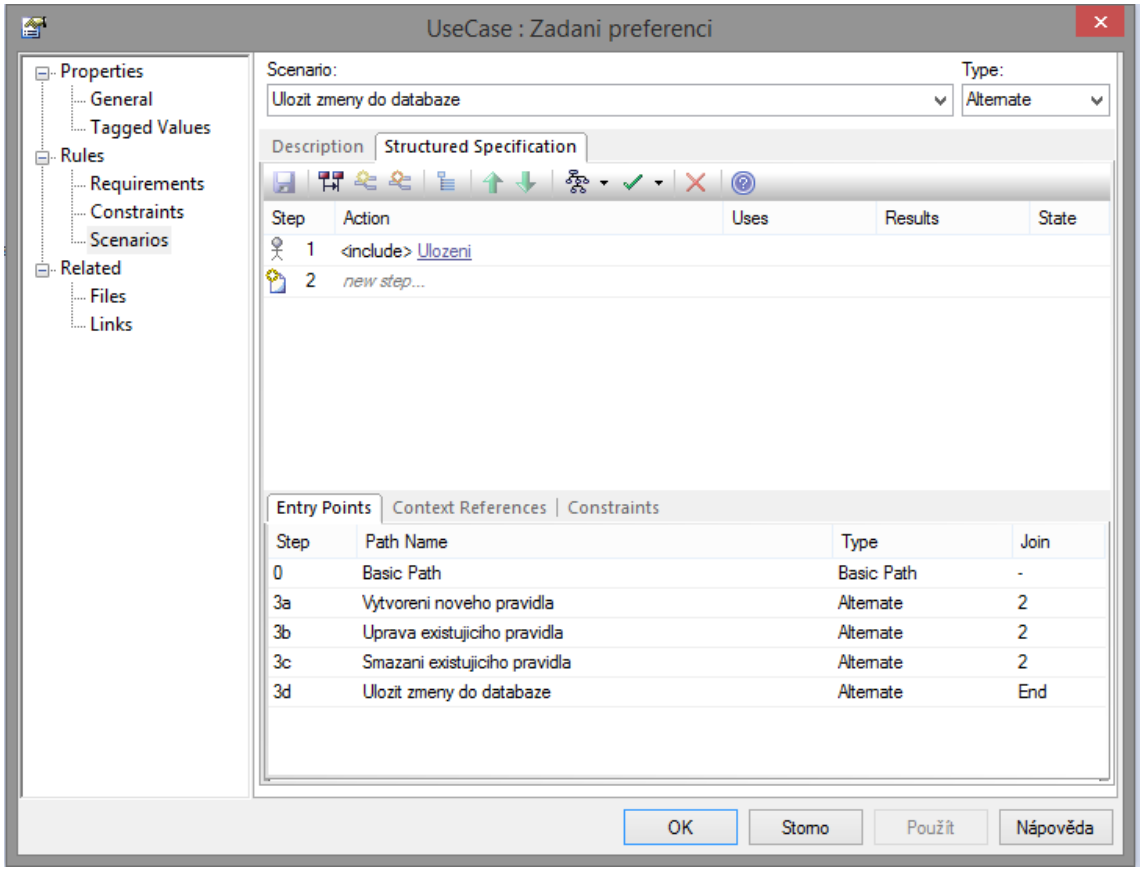
Obrázek 9 Scénář: Zadání preferenci – Vytvoření nového pravidla



Obrázek 10 Scénář: Zadání preferencí - Úprava existujícího pravidla



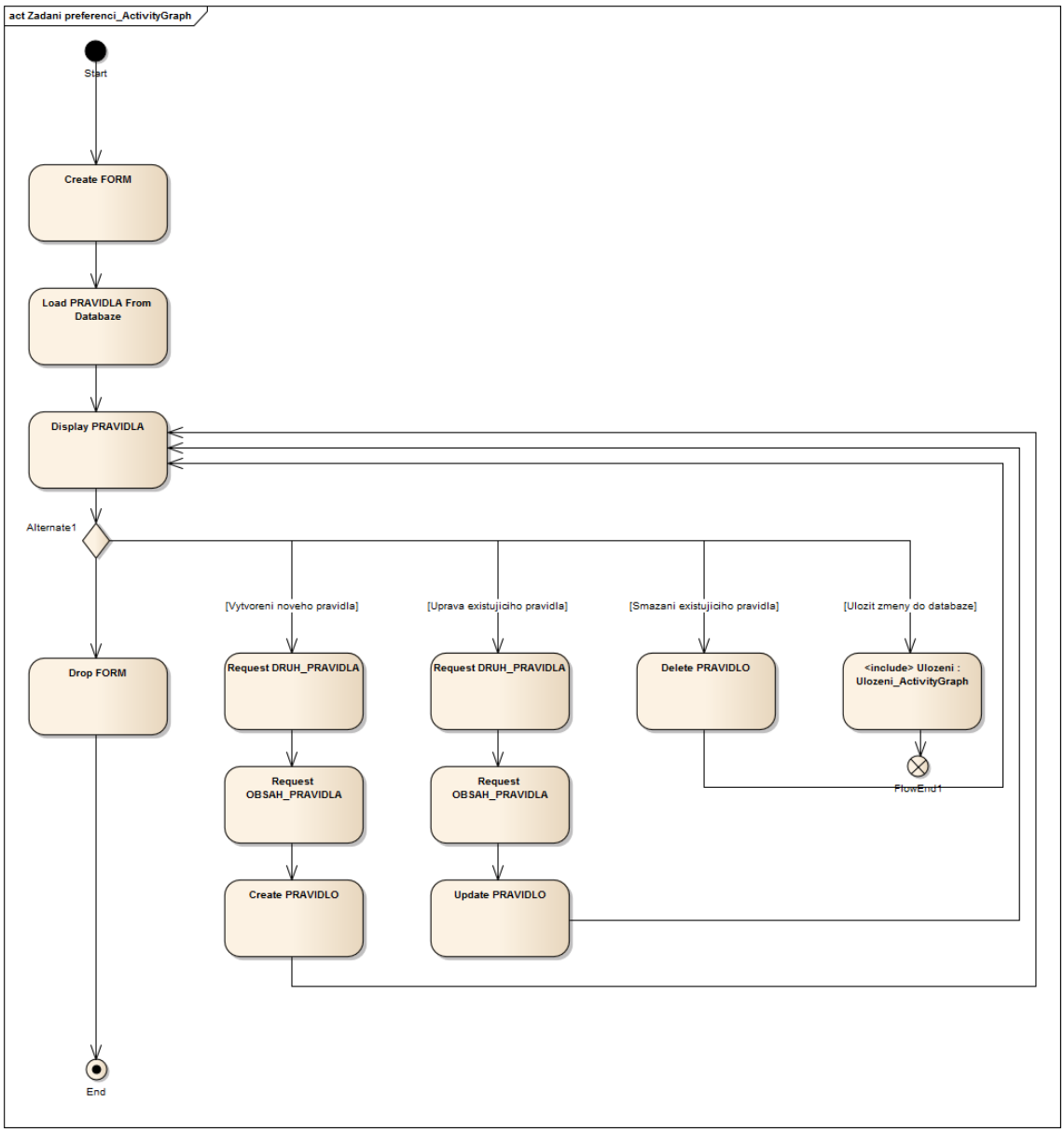
Obrázek 11 Scénář: Zadání preferencí - Smazání existujícího pravidla



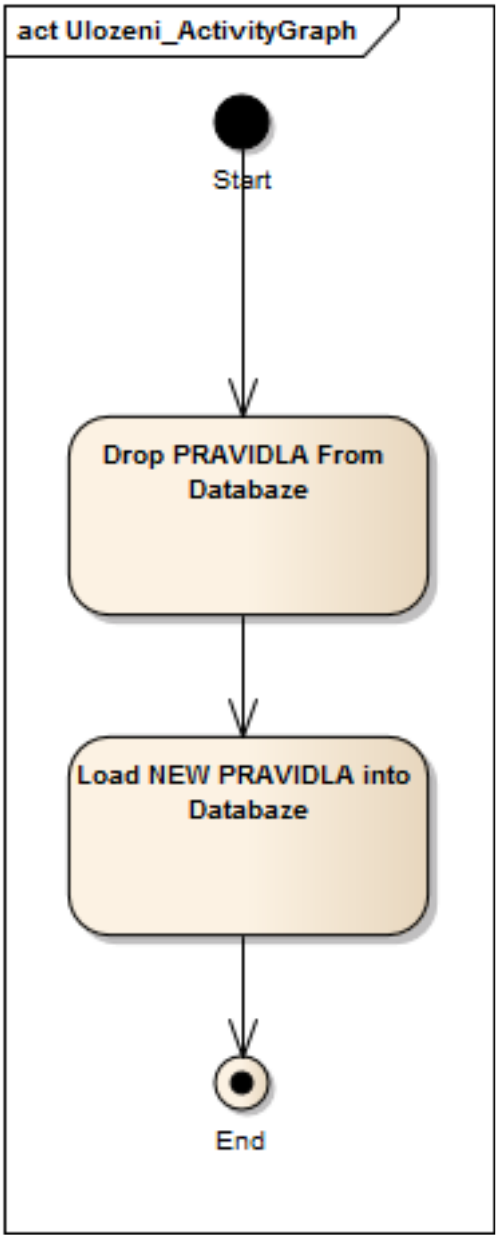
Obrázek 12 Scénář: Zadání preferencí - Uložit změny do databáze

2 Diagram aktivit

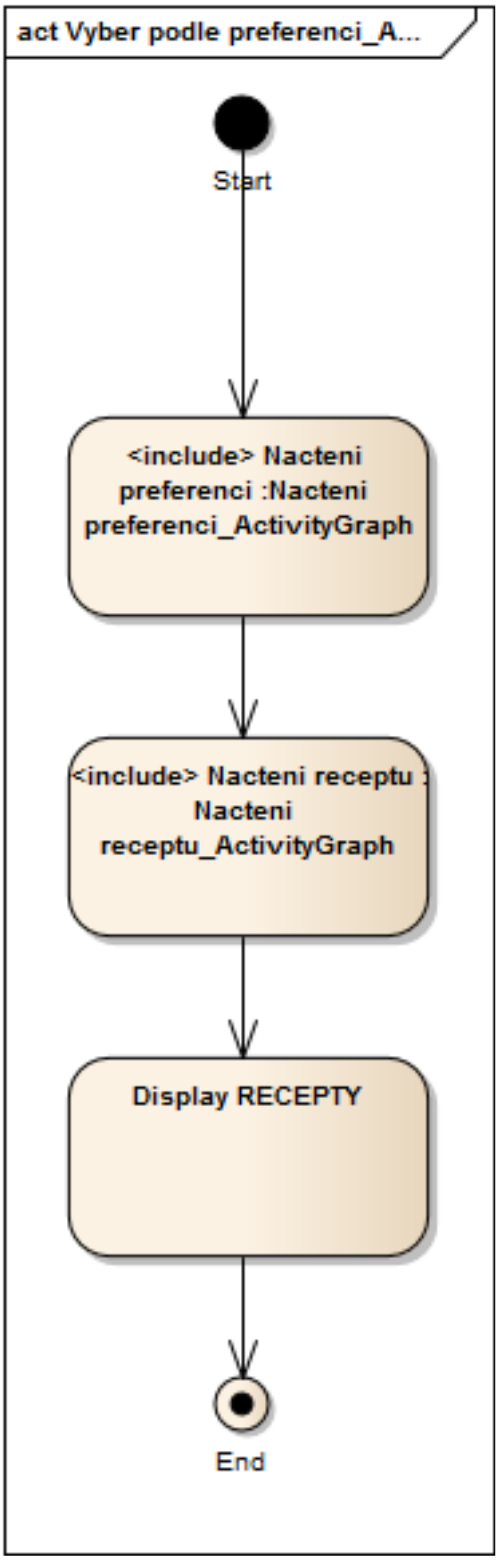
Diagram aktivit se používá pro modelování procedurální logiky, procesů a zachycení postupů. V našem případě je hlavní diagram Zadání preferencí, který popisuje vytváření, úpravy, mazání a ukládání pravidel, které jsou používány v našeptávači. Ostatní níže zobrazené diagramy popisují pouze dílčí části, kde se popisuje ukládání pravidel, výběr a načítání receptů z databáze.



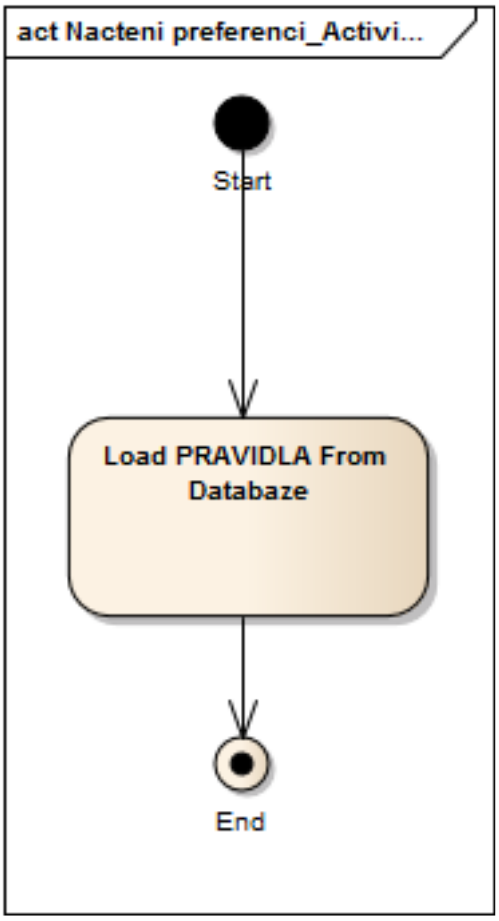
Obrázek 13 – Diagram aktivit: Zadání preferencí



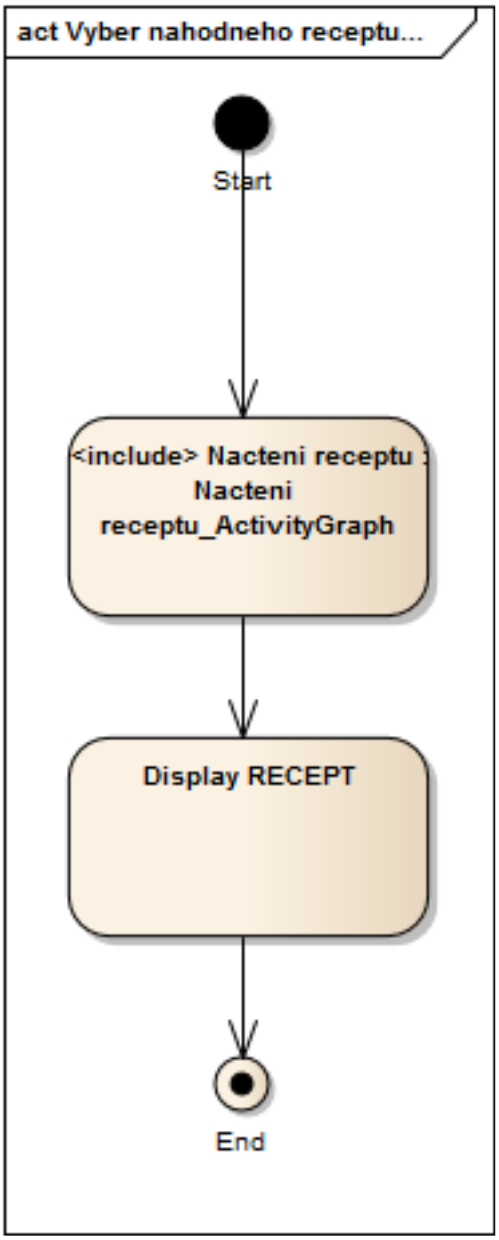
Obrázek 14 – Diagram aktivit: Uložení



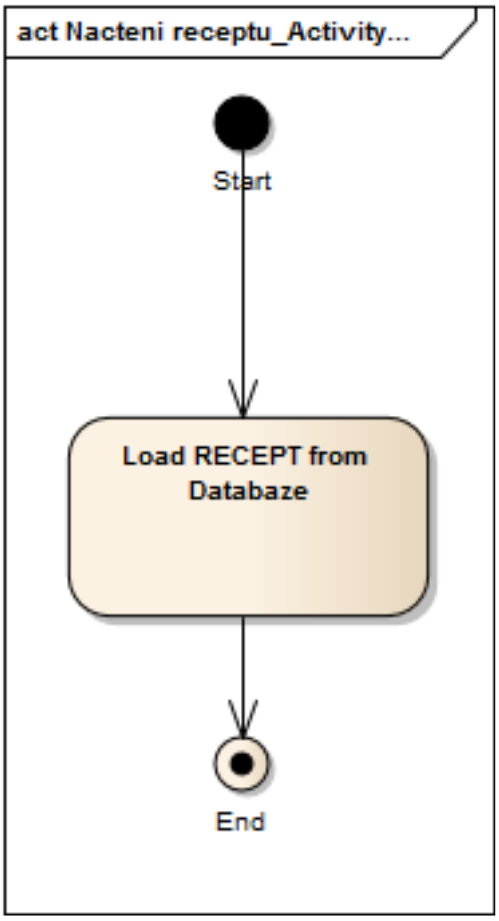
Obrázek 15 – Diagram aktivit: Výběr podle preferencí



Obrázek 16 – Diagram aktivit: Načtení preferencí



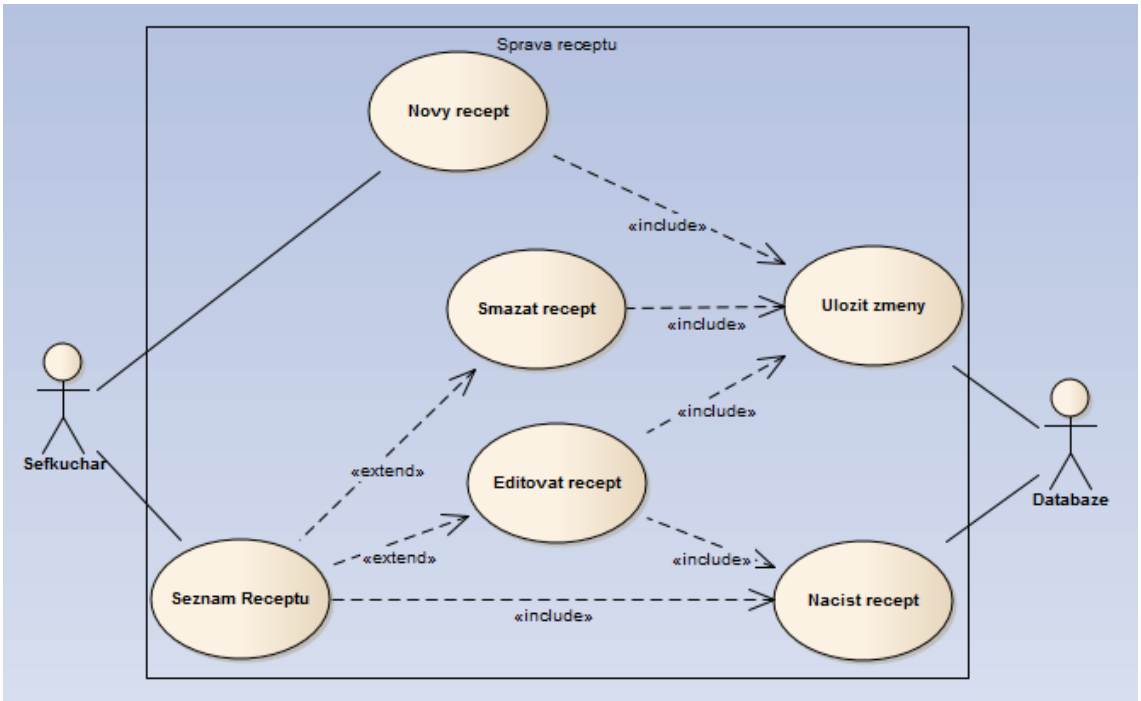
Obrázek 17 – Diagram aktivit: Výběr náhodného receptu



Obrázek 18 – Diagram aktivit: Načtení receptu

3 Diagram tříd

Diagram tříd byl sestaven na základě Use Case diagramu jiné skupiny.

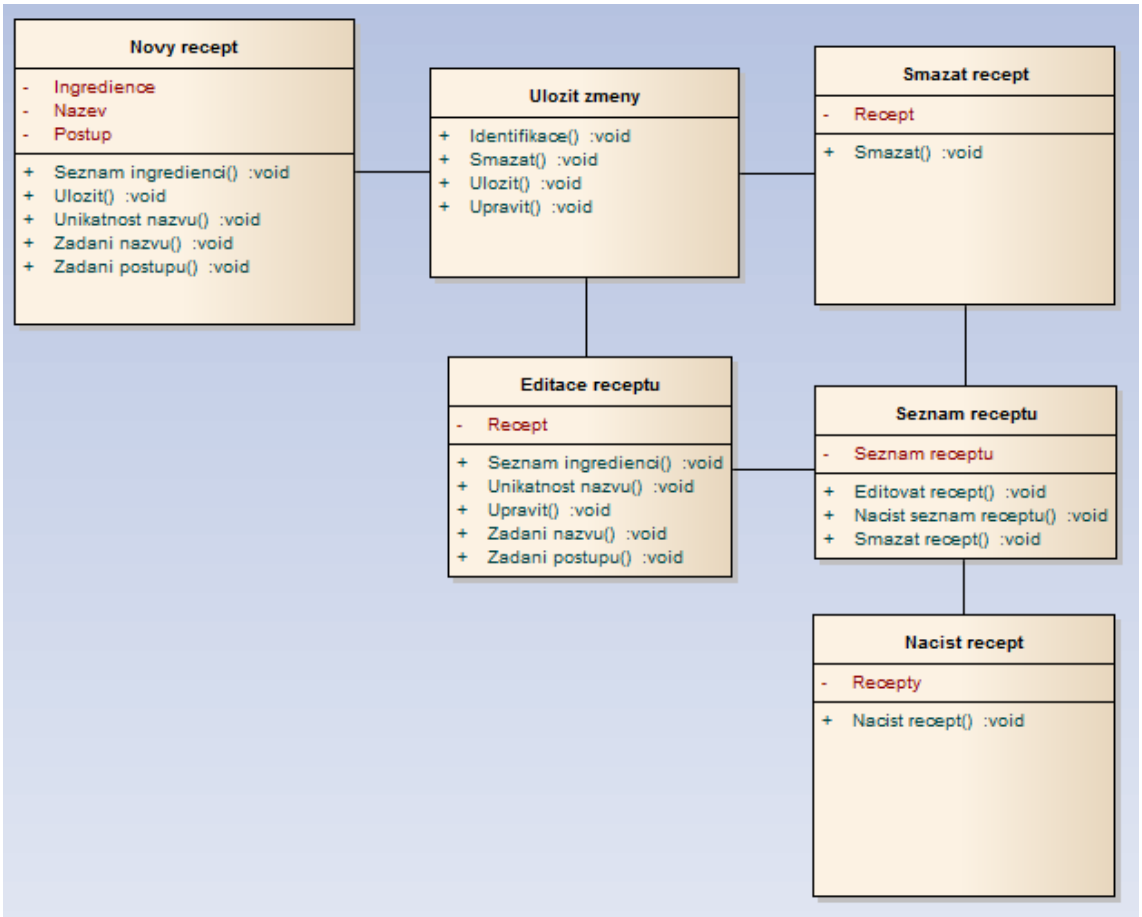


Obrázek 19 – Use Case: Správa receptu

Z Use Case diagramu jde vidět, že se jedná o základní manipulaci s recepty, jako je vytvoření nového receptu, smazání, editace a zobrazení seznamu receptů.

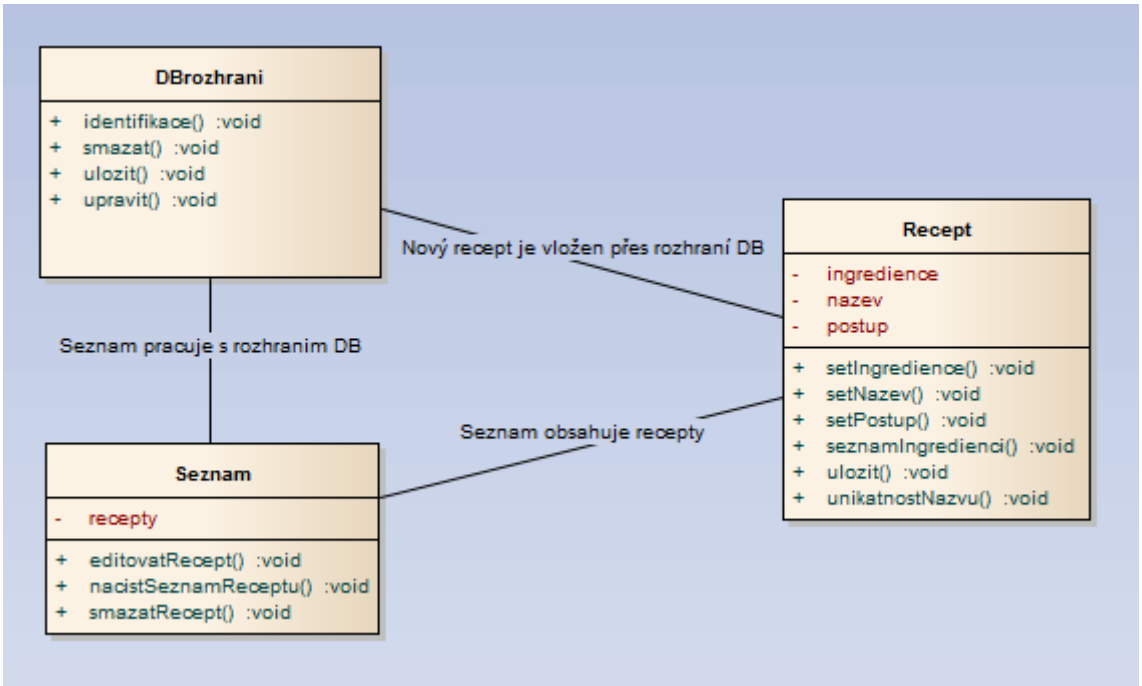
Pro lepší porozumění danému problému jsme si pomohli náhledem do scénářů, které byly napsány přehledně a srozumitelně, nicméně u S*mazat recept* a *Nacist recept* scénáře chyběly. Přesto pro nás nebyl problém sestavit diagram tříd a následně i sekvenční diagram.

U prvotního návrhu diagramu tříd jsme se dopustili špatných názvů tříd a nepopsali jsme vazby, ale logická stavba byla správně, diagram vypadal následovně.



Obrázek 20 – Původní diagram tříd

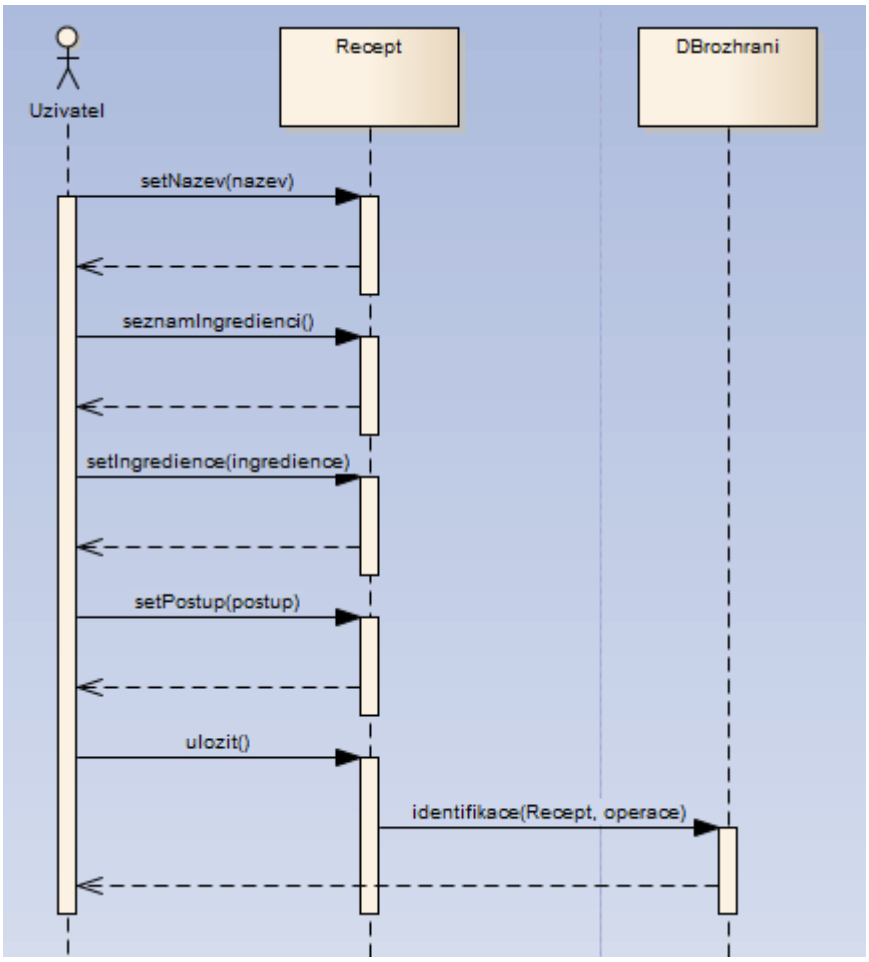
Hned v počátku tvorby sekvenčního diagramu jsme si uvědomili, že jsme v podstatě věrně přenesli jednotlivé use case diagramy i do tříd, takže každý use case byl reprezentován svojí vlastní třídou. Což není uplně vhodné a celý diagram jsme zjednodušili do následující podoby.



Obrázek 21 – Upravený diagram tříd

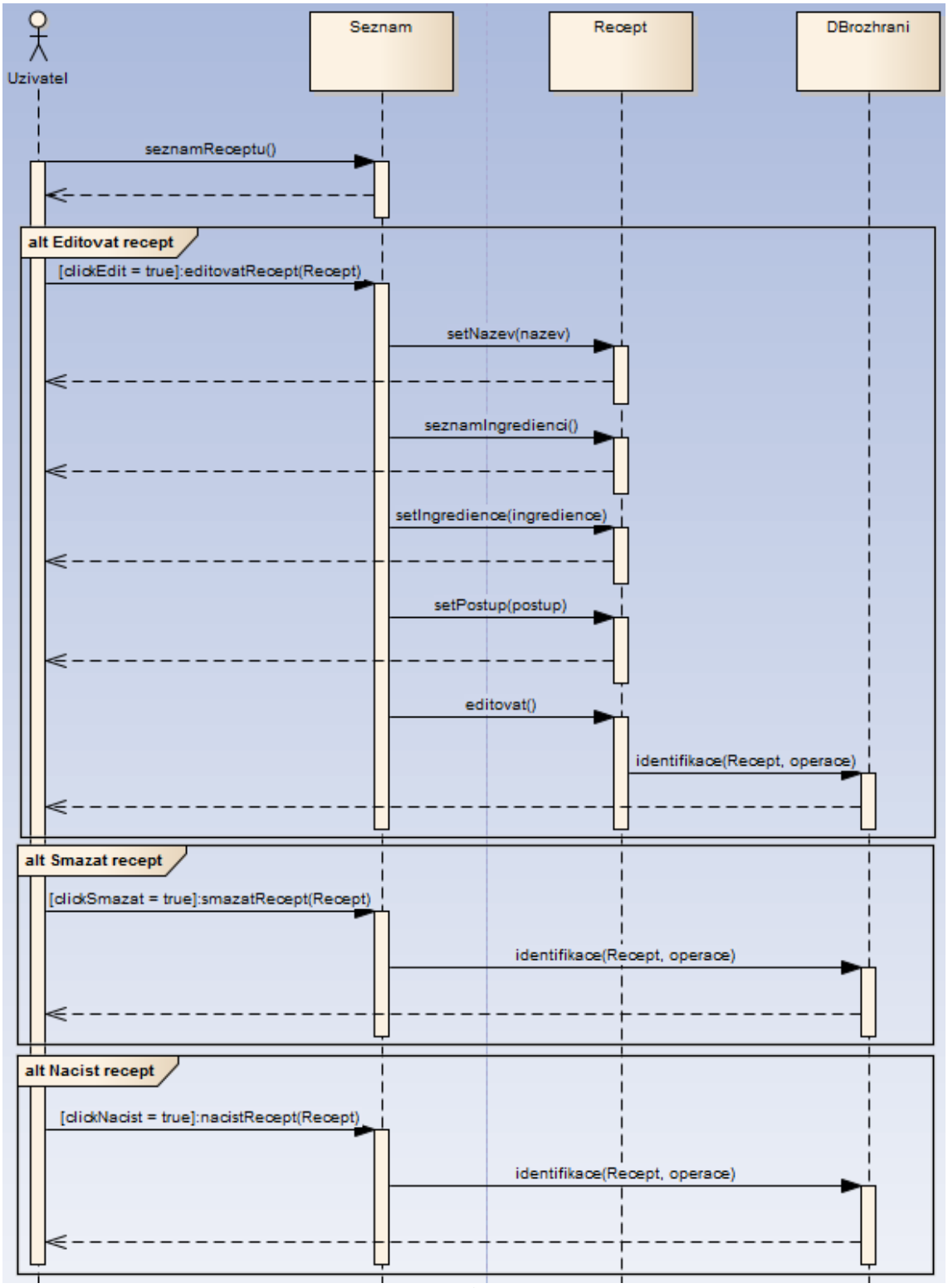
4 Sekvenční diagram

První ze sekvenčních diagramů zachycuje vytvoření nového receptu. Pokud chce uživatel vytvořit nový recept, tak na objektu *Recept* volá následující metody. Nejdříve určí název receptu pomocí metody *setNazev*, která zkontroluje unikátnost názvu. Dále se uživateli zobrazí seznam ingrediencí (*seznamIngredienci*), z kterého vybere potřebné ingredience (*setIngredience*). Na závěr uživatel uloží recept, který je poslán v metodě *indentifikace* objektu *DBrozhrani*, která identifikuje, že se jedná o vložení nového receptu a vloží jej do databáze.



Obrázek 22 – Sekvenční diagram: Vytvoření nového receptu

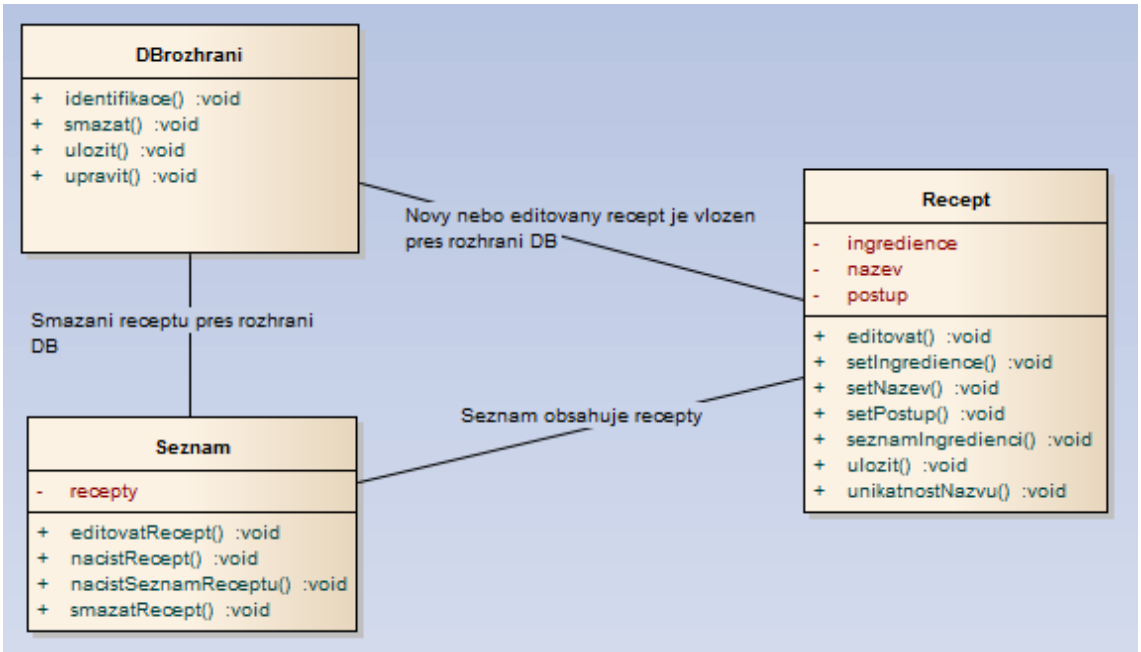
Další diagram zachycuje zobrazení seznamu receptu a příslušně manipulace s recepty. Součástí diagramu jsou další fragmenty a to konkrétně editace recepte, smazání recepte a načtení receptu. Uživateli je nejdříve zobrazen seznam receptů a poté se může rozhodnout, zda chce určitý recept zobrazit, smazat nebo editovat.



Obrázek 23 – Sekvenční diagram: Seznam receptů

5 Upravený diagram tříd

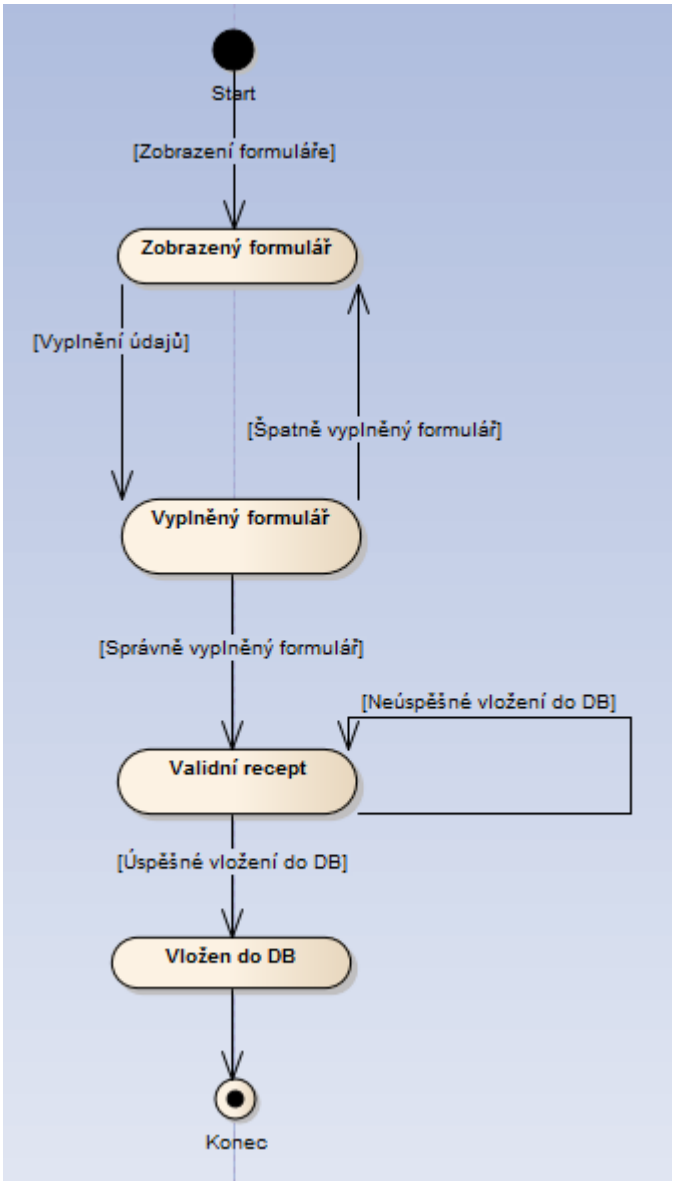
Při tvorbě sekvenčních diagramů jsme si uvědomili, že nám chybí metoda *nacistRecept* u třídy *Seznam* a metoda *editovat* u třídy *Recept*.



Obrázek 24 – Upravený diagram tříd podle sekvenčních diagramů

6 Stavový diagram

Stavový diagram pro vložení nového receptu. Uživatel vyplní formulář, pokud jsou údaje validní, je možné recept přidat do databáze.



Obrázek 25 – Stavový diagram