TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# Software pro správu projektu

## Semestrální práce

*Studijní program:* B0613A140005 – Informační technologie

*Autor práce:* **Kevin Daněk**
*Vedoucí práce:* doc. Ing. Jiřina Královcová, Ph.D.

Liberec 2022

# Software pro správu projektu

## Abstrakt

Cílem semestrální práce bylo navrhnout a realizovat informační systém v jazyce Java.

**Klíčová slova:** Java

## Abstract

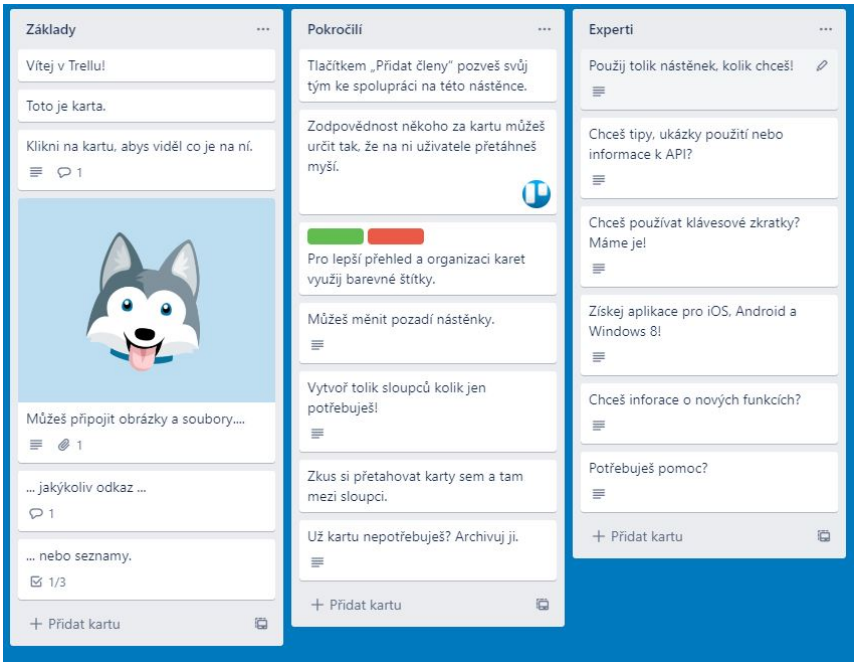
This report describes the `tulthesis` package for Technical university of Liberec thesis typesetting using the LaTeX typographic system.

**Keywords:** LaTeX, class, TUL

# Obsah

# 1 Zadání

Cílem práce je vytvořit program, který bude realizovat správu vývoje softwaru pomocí metody Kanban. Metodou Kanban rozumíme rozdělení práce na menšéíé úkoly, které se postupně přesouvají vývojem z různých sloupců až do stavu, kdy jsou označeny za hotovy.




Obrázek 1.1: Ukázka vizuální reprezentace metody Kanban (Ze služby Trello)

Program by měl být schopný

- **vytvořit a spravovat** vícero nástěnek
- **přidávat a spravovat sloupce** v nástěnkách
- **přidávat a spravovat úkoly** ve sloupcích
- **přidávat a přidělovat** zaměstnance

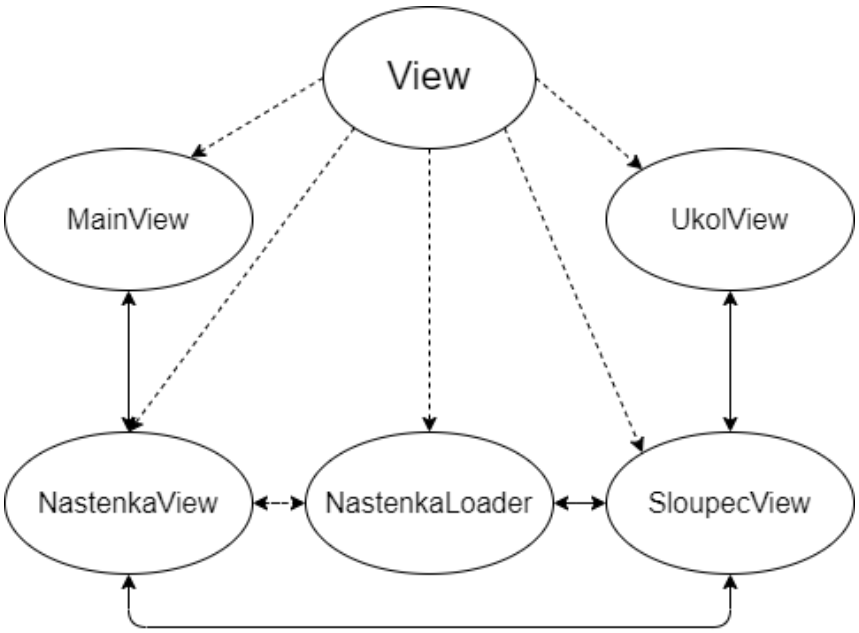
# 2 Specifikace implementace

## 2.1 Formát dat

Program ukládá svá data do souborů formátu JSON. Každý klíč (až na vyhrazený klíč `class`) odpovídá atributu objektu (instance třídy), kde hodnota představuje okamžitou hodnotu v době uložení na disk.

## 2.2 Závislost tříd

Architektura programu stojí na zjednodušeném MVC modelu, kde *model* představují entity balíku `trello`, *controller* třídy balíku `stores` a *view* balík `ui`. Níže na obrázku lze vidět, jak se uživatel může pohybovat mezi pohledy. Plná čára značí krok, přerušovaná závislost, resp. dědičnost.




Obrázek 2.1: Závislosti a cesty mezi jednotlivými pohledy uživatelského rozhraní

## 2.3 Kompetence tříd

### 2.3.1 Podbalík `trello`

Třídy balíku `trello` reprezentují jednotlivé entity programu a slouží jako nositelé dat.

### 2.3.2 Podbalík `stores`

Balík `stores` slouží pro třídy, které aplikace používá pro ukládání dat do paměti a pro jednotný přístup k těmto datům. Tyto třídy jsou `singletony`, které uchovávají instance jednotlivých entit.

#### Třída `NastenkaStore`

Třída `NastenkaStore` je `singletonem`, který uchovává a poskytuje přístup k instanci třídy `Nastenka`. Umožňuje načíst, uložit a vypsat aktuální instanci ze souborového systému.

### 2.3.3 Podbalík `ui`

Podbalík `ui` v sobě uchovává uživatelské rozhraní, které je ovládáno skrz standardní vstup a výstup.

#### Třída `TextMenu`

Třída `TextMenu` slouží jako stavební kámen pro obyčejná textová rozhraní s očíslovanými možnostmi. Instance této třídy si uchovává dva seřazené seznamy (třídy `LinkedList`):

1. Seznam textových řetězců, který uchovává **nadpisy možností**, a
2. Seznam spustitelných funkcí, který uchovává **obslužné metody k těmto možnostem**.

Vzhledem k tomu, že je potřeba vytvořit vztah mezi těmito seznamy, nabízí se k diskuzi implementace datové struktury `Map`. Já jsem ovšem chtěl zachovat pořadí, ve kterém se prvky přidávají, a usoudil jsem, že se mi s instancemi třídy `LinkedList` bude pracovat lépe.

Třída poskytuje možnost přidávat možnosti a ty jsou následně očíslovány podle toho, v jakém pořadí byly přidány. Zároveň dovoluje volat obslužné metody a menu naformátovat (viz programátorská dokumentance).

#### Třída `View`

Třída `View` slouží jako rodičovská třída pro tzv. `pohledy rozhraní` - to jsou třídy reprezentující jednotlivé části UI. Každý pohled je potomkem této třídy a využívá jejich zděděných prostředků.

Tato třída nabízí přístup k

- lokalizovaným zdrojům,
- instanci třídy `Scanner`
- metodě pro pozastavení programu do stisknutí klávesy `enter`
- a metodě pro uložení aktuálně načtené nástěnky

#### Třída `MainView`

Třída `MainView` je vstupním bodem programu. Třída používá instanci třídy `TextMenu` k vykreslení jednoduchého menu pro otevření nástěnky, změnu jazyka či otevření průzkumníka souborů v adresáři s daty.

#### Třída `NastenkaView`

Třída `NastenkaView` je pohled rozhraní pro práci s nástěnkou. Dovoluje pomocí menu procházet sloupce aktuální nástěnky, její zaměstnance či ji manuálně uložit.

#### Třída `NastenkaLoader`

Třída `NastenkaLoader` je pohled rozhraní, který se spoustí v případě, kdy není žádná nástěnka načtena, nebo uživatel vybere načtení jiné nástěnky. Tato třída také umožňuje vytvoření nové nástěnky.

#### Třída `SloupecView`

Třída `SloupecView` je pohledem rozhraní, který slouží ke správě sloupce v nástěnce. Umožňuje přidávat, upravovat, odebírat a vypisovat úkoly.

#### Třída `UkolView`

Třída `UkolView` je pohledem zobrazení, který se ukazuje při úpravě úkolu. Umožňuje k úkolu přiřadit zaměstnance, změnit termín či prioritu.

### 2.3.4 Podbalík `lib`

#### Třída `FileSystemUtils`

Třída `FileSystemUtils` je knihovní třídou, která slouží k poskytování služeb spojených se soubory a souborovým systémem. Jedná se o rozhraní, které poskytuje metody pro korektní přístup ke zdrojům, které program potřebuje. Zároveň tato třída poskytuje konstanty pro cesty k důležitým souborům a adresářům.

### Třída `StringUtils`

Třída `StringUtils` je knihovní třídou, která poskytuje pomocné metody pro práci s textovými řetězci. Jedná se např. o metody pro generování vertikálního odsazení, zalomení řádku či jeho zkrácení.

### Třída `ScannerUtils`

Třída `ScannerUtils` je knihovní třídou poskytující metody, které jsou nadstavbou pro standardní metody třídy `Scanner`. Jedná se například o metodu `nextDate`, která slouží ke korektnímu načtení data (datum), či metoda poskytující opětovné načtení dat v případě chyby (`nextDataUntilValid`).

### Třída `ObjectUtils`

Třída `ObjectUtils` je knihovní třídou, která slouží k poskytování služeb tématicky se týkajících instancí objektů. Důvodem vzniku této třídy byla potřeba po prostředku, který dokáže dynamicky vytvořit mapu mezi `atributy` a jejich `gettery/settery`.

### Třída `JsonUtils`

Třída `JsonUtils` je knihovní třídou pro obousměrnou konverzi instancí tříd do JSON objektů a naopak. Třída je designována tak, aby možnost této konverze výrazne nepodmiňovala návrh jednotlivých tříd.

Pro převod instance třídy do JSON objektu slouží metoda `encodeObjectToJSON`, která jako argument přijímá objekt k převedení. Metoda pomocí třídy `ObjectUtils` obdrží slovník atributů a jejich `getterů`. Následně metoda iteruje skrz záznamy tohoto slovníku a jednotlivé `gettery` volá vůči instanci třídy, která byla předána metodě v argumentu.

Pokud je návratová hodnota typu, který je potomkem třídy `Collection`, pak se metoda encodeObjectToJSON zavolá rekurzivně pro každý prvek kolekce, přičemž kolekce bude převedena na typ `JSONArray`.

Pro převod z JSON objektu zpět na instanci třídy je použit podobný proces, pouze místo `getterů` si metoda `parseJsonObject` vyžádá mapu atributů a jejich `setterů`. Pro to, aby metoda věděla, jakou třídu má instancovat, je při ukládání objektu do JSON objektu uloženo kanonické jméno třídy (pod klíčem `class`).