FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Systém pro tvorbu automatického generátoru zpráv ze strukturovaných dat

Jan Ulrych

PLZEŇ

2023

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Systém pro tvorbu automatického generátoru zpráv ze strukturovaných dat

Jan Ulrych, Ing. Jakub Sido

Vedoucí práce
Ing. Jakub Sido

PLZEŇ **2023**

Citace v seznamu literatury:
ULRYCH, Jan. *Systém pro tvorbu automatického generátoru zpráv ze strukturovaných dat*. Plzeň, 2023. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Jakub Sido.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan ULRYCH**
Osobní číslo:	**A20B0264P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Systém pro tvorbu automatického generátoru zpráv ze strukturovaných dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte problematiku generování novinových zpráv šablonovým způsobem.
2. Navrhněte systém umožňující přívětivou tvorbu šablon a server pro pravidelnou publikaci z nich generovaných novinových zpráv.
3. Navržený systém implementujte.
4. Úplnost systému otestujte přenesením služby generující novinové zprávy burzovního zpravodajství, která vznikla v projektu TAČR-TL02000288.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Ing. Jakub Sido**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **3. října 2022**
Termín odevzdání bakalářské práce: **4. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2022

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 4. května 2023

. . . . . .
Jan Ulrych

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Cílem této bakalářské práce je vytvořit systém pro tvorbu automatického generátoru zpráv ze strukturovaných dat za použití uživatelsky definovaných šablon. Práce zkoumá problematiku automatického generování zpráv. Na základě zkušeností novinářů s existujícím systémem pak formuluje požadavky na nový systém. Na základě těchto požadavků je pak navržen a implementován systém, který uživatelům dovoluje definovat šablonu generované zprávy, stahovat data z různých zdrojů a na základě definovaných šablon v předem daný čas automaticky generovat text.

Abstract

The aim of this bachelor thesis is to develop a system for creating an automatic report generator from structured data using user-defined templates. The thesis explores the issue of automatic message generation. It then formulates requirements for a new system based on journalists experience with an existing system. Based on these requirements, a system that allows users to define a template for the generated report, download data from various sources, and automatically generate text based on the defined templates at a predefined time, is then designed and implemented.

Klíčová slova

novinový článek • automatizace • šablony • software • automatické generování • systém

Poděkování

Chtěl bych tímto poděkovat vedoucímu této práce, panu Ing. Jakubu Sidovi, za příkladné vedení bakalářské práce, všechny připomínky, které napomohly vylepšení důležitých vlastností systému a asistenci při replikování funkcionality existujícího systému. V neposlední řadě patří poděkování mé rodině a přátelům, kteří mě podporovali po celou dobu studia a při psaní této práce.

Obsah

Úvod 1

Cílem této práce je vytvořit systém, který umožní automatické generování zpráv ze strukturovaných dat dle uživatelsky definovatelných šablon. Systém by měl novinářům usnadnit práci při psaní textů, které se často opakují, a zároveň jim nad generovanými texty ponechat plnou kontrolu. Díky tomu by se tak novináři mohli věnovat práci, do které mohou vložit více vlastní kreativity.

První část této práce se zabývá problematikou automatického generování zpráv, popisuje práci novinářů a jejich zkušenost s automatickým generováním zpráv pomocí stávajícího systému a popisuje požadavky na nový systém.

Další část práce se zabývá návrhem systému. Navrhuje řešení jednotlivých problémů, popisuje technologie, které je možné k jeho implementaci použít, a podle zjištěných požadavků navrhuje jednotlivé vlastnosti systému. Tato část také navrhuje databázový model a webovou a serverovou část.

V praktické části práce popisuje implementaci jednotlivých částí systému. Popisuje zvolené technologie, důležité vlastnosti těchto technologií, využití těchto vlastností při vývoji aplikace a konkrétní implementaci systémových požadavků, jejichž řešení bylo navrženo v předchozí části. Tato část také popisuje zvolený způsob distribuce systému a jeho instalaci.

Poslední část práce se pak zabývá přenesením funkcionality z existujícího systému do systému nového, testováním zdrojového kódu a efektivitou generování zpráv pomocí nového systému.

2 Problematika generování novinových zpráv

Generování zpráv ze strukturovaných dat je proces, při kterém se z formátovaných a organizovaných dat vytváří textová zpráva. Tento proces je používán pro automatické vytváření zpráv, které shrnují důležité informace obsažené v těchto datech. Hlavním cílem tohoto postupu je zobrazení dat v přehledné a snadno čitelné formě.

Tato problematika se momentálně řeší hlavně v oblasti umělé inteligence, kde systémy musí na základě trénovacích dat a požadavků na výslednou zprávu vytvořit takovou zprávu, která bude co nejvíce připomínat něco, co by napsal člověk. V době automatizace a požadavku na co nejrychlejší a pokud možno nejobsáhlejší informace se tato problematika řeší v mnoha oblastech, jednou z nich je i oblast novinářská.

2.1 Existující systém

Ve světě již existuje mnoho novinářských domů, které využívají systémy pro automatické generování novinových zpráv, jako například *Associated Press*, *Forbes*, *ProPublica* nebo *Los Angeles Times* [1]. V České republice tato technologie zatím příliš využívána není. Důvodem může být menší trh v porovnání s vysokými náklady na realizaci takových systémů.

V rámci projektu TAČR-TL02000288 vznikl systém, který pomocí automatizace každý den po uzavření burzy generuje novinové články a sumarizuje obchodování na burze v uplynulém dni. Tento systém byl využíván serverem e15.cz v rubrice Burzovní robot[1].

[1] Odkaz na rubriku Burzovní robot: `https://www.e15.cz/tag/burzovn%C3%AD%20robot/1`.

2.1.1 Jak novináři pracují

Novináři označují psaní reportů o obchodování na pražské burze jako víceméně rutinní záležitost, do které novináři nemohou vnést příliš své vlastní kreativity a originality. Z tohoto pohledu by byla automatizace tvorby článků o tomto tématu ideální. Novináři každý den sledují vývoj burzy na webových stránkách *Prague Stock Exchange* a následně píší články na základě informací z těchto stránek. PSE publikuje data o změnách na burze se zpožděním 15 minut, což vede k situacím, kdy novináři občas musí své předchozí zprávy dementovat a opravovat [2].

2.1.2 Očekávání novinářů

Ze studie v rámci projektu vyplynulo, že od automatizace novináři očekávají zejména zrychlení publikace článků. Z pozorování vyplynulo, že zatímco novinář napíše titulek zhruba do 30 minut po uzavření burzy, při použití automatizace je tento titulek vydán přibližně o 15 minut dříve. Podobně je tomu tak i při vydávání rozsáhlejších článků o burze, v tomto případě ale ruční psaní článku zabere cirka 60 minut. Protože je stále nutná kontrola a případně úprava titulků a článků editory, není možné vydávat články ihned po jejich vygenerování počítačem. V případě, že by kontrola a úprava nutná nebyla, bylo by možné články vydávat ihned po jejich vygenerování, tedy ihned po uzavření burzy a publikování dat o burze PSE.

2.1.3 Poznatky novinářů

Z poznatků od novinářů je patrné, že jsou spokojeni s rychlostí vydání nového článku. Ten je zpravidla vydáván ihned po uzavření burzy a jeho zkontrolování editorem. V případě ručního psaní článků by dosáhnutí takové rychlosti vydání článku nebylo možné. Na druhou stranu by novináři chtěli mít větší kontrolu nad generovaným textem. Z generovaných zpráv je totiž díky občasným nedokonalostem a nepřesnostem patrné, že nebyly psány lidskou rukou, ale umělou inteligencí, která nemá takový smysl pro psaný text jako novinář s mnohaletou praxí. Automatické generování článků dává novinářům větší volnost v psaní článků o tématech, ve kterých mohou více projevit svoji kreativitu a originalitu. Psaní článků o obchodování na burze sice může být rutinní záležitost, ale zabere poměrně hodně času a novinář v ní nemá větší možnost seberealizace.

2.1.4 Použití šablonového generování

Automatické generování zpráv s použitím šablon bylo v Česku poprvé použito Českou tiskovou kanceláří v roce 2018 při obecních a senátních volbách [3]. Po zveřejnění dat Českým statistickým úřadem byla do předem připravené statické šablony vložena data a z nich byla vytvořena zpráva. Ta byla pouze zkontrolována editorem

a následně publikována. Takových zpráv bylo publikováno zhruba 200. Novináři si pochvalovali hlavně ušetřený čas, který mohli věnovat ohlasům a dalším zajímavostem z voleb.

2.1.5 Výhody a nevýhody automatického generování zpráv za použití šablon

Hlavní výhodou automatického generování zpráv ať už za použití šablon či umělé inteligence je rychlost. Článek, který by novináři mohl zabrat hodinu, může být vygenerován za zlomek sekundy. Další výhodou je dobrá testovatelnost systému. Každá šablona se na různých datech dá dobře otestovat a novináři mají jistotu, že je výsledek pokaždé deterministický. To je i hlavní výhoda šablonového přístupu oproti generování textů za použití umělé inteligence. Nevýhodou je to, že novináři často nejsou schopni popsat tvorbu novinového článku jednou šablonou. Různí novináři se ve své tvorbě liší, data reprezentují jinak (např. růst o 3 % jeden novinář popíše jako výrazný a druhý novinář jako velmi výrazný) a zprávy tak logicky od každého novináře vypadají jinak. Šablony by však mohly styl článků sjednotit a čtenář by tak pokaždé přesně věděl, co má od článku čekat a kde najít informace, které jsou pro něho důležité.

2.2 Požadavky na nový systém

Nový systém by měl uživatelům dávat možnost definovat strukturu novinového článku nebo jiné automaticky generované zprávy tak, aby měl uživatel vždy stoprocentní jistotu, že výsledný text bude pro vstupní data vypadat přesně tak, jak si jej nadefinoval. Zároveň by měl systém definované zprávy v předem stanovených časech automaticky generovat a výsledky těchto generování poskytovat jak v grafické podobě, tak i pomocí aplikačního rozhraní, aby bylo umožněno navázání nového systému na systémy stávající.

2.2.1 Zdroje dat

Systém by měl uživateli dovolit používat data z různých zdrojů. Pokud bude uživatel chtít získat data pro generování zprávy například pomocí FTP, nebo voláním webového API, mělo by mu to být umožněno. Stejně jako by měl systém umožnit použití různých datových zdrojů, měl by umožnit použití i různých datových formátů. Ať už se tedy uživatel rozhodne použít například data ve formátu `CSV` nebo `JSON`), systém by měl data zpracovat a použít pro generování textu.

2.2.2 Transformace dat

Systém by měl umožnit data, která se uživatel rozhodne pro generování zprávy použít, libovolně transformovat a provádět nad nimi výpočty. Data, která vzejdou z těchto transformací, by pak měla být použita pro generování zprávy.

2.2.3 Definice šablon

Systém by měl umožňovat uživatelům definovat šablony (návody), podle kterých se výsledné zprávy budou generovat. Rozhraní pro definici šablon by mělo být uživatelsky přívětivé, zároveň by však mělo dovolovat definování komplexních a robustních šablon.

Tvorba šablony by měla v podstatě odpovídat tomu, jak se novinář na základě jeho pozorování burzy rozhoduje, co do výsledného článku napíše. Pokud si tedy novinář při psaní nadpisu článku pokládá otázku „Roste dnes burza výrazně nebo jen mírně?“ a odpovědí na tuto otázku je „Burza roste jen o 0.1 procenta, tedy mírně.“, pak by se měl při definici šablony odehrávat podobný proces. Novinář by tedy měl mít možnost v šabloně nadefinovat podmínku pro aktuální růst burzy a pro různé výstupy této podmínky do šablony vložit příslušný text.

Návrh systému 3

3.1 Řešení různorodosti a transformace dat

Systém musí vyřešit požadavek na různorodost dat a jejich transformaci. Tento problém bude řešen za pomoci uživatelsky definovatelných skriptů, které uživateli dovolí získat data z jakéhokoli zdroje a v jakémkoli formátu. V rámci skriptu pak bude moci uživatel získaná data jakkoli transformovat. Skript bude spuštěn vždy na začátku celého procesu generování zprávy.

3.1.1 Předzpracování dat

Data stažená a zpracovaná pomocí skriptu bude možné ještě před samotným generováním zpracovat a analyzovat pomocí jednoduchého nástroje. Ten umožní uživateli vypočítat například průměr z dat, jejich součet, nebo spočítat počet výskytů určité hodnoty. Výsledky těchto akcí budou poté dočasně uloženy pouze pro potřeby daného generování. Kdybychom tyto akce prováděli v rámci skriptu, mohlo by to způsobit zbytečné redundantní hodnoty v datovém souboru, proto jsou tyto akce od skriptu odděleny.

3.2 Návrh šablon

Systém bude umožňovat uživatelům definovat šablony (návody), podle kterých se dále budou generovat zprávy. Šablona bude obsahovat název, skript vykonávaný před generováním zpráv, seznam akcí pro předzpracování dat, soubor s testovacími daty a nakonec jednotlivá primitiva pro popis struktury zprávy generované touto šablonou. Primitiva by měla odpovídat různým procesům při ručním psaní textu. Například „procházím všechna data a něco v nich hledám“ nebo „na základě dat budu psát o něčem“ atd.

3.2.1 Primitiva šablony

Šablony budou obsahovat následující primitiva (v grafickém rozhraní budou tato primitiva prezentována jako bloky): text, odstavec, seznam bloků, podmínka, cyklus, přepínač, randomizér a odkaz.

3.2.1.1 Text

Textové primitivum bude uživateli umožňovat definovat text, který bude zpráva obsahovat. V tomto textu budou obsaženy zástupné symboly, které budou při generování nahrazeny skutečnými daty.

3.2.1.2 Odstavec

Odstavec bude mít stejnou funkci jako text s tím rozdílem, že na jeho konci dojde k automatickému vložení nové řádky do generované zprávy.

3.2.1.3 Seznam bloků

Seznam bloků bude sloužit pouze jako kontejner na ostatní bloky. Bude umožňovat uživateli jednodušší orientaci v šabloně. Jednotlivé seznamy bloků budou moci být pro větší přehlednost pojmenované. Toto pojmenování ani příslušnost bloků v seznamu bloků nijak neovlivní generovanou zprávu.

3.2.1.4 Podmínka

Podmínka bude sloužit k větvení při generování zpráv. Podmínka bude odpovídat struktuře `if-else-if` známé z většiny programovacích jazyků. Bude tedy umožňovat uživateli definovat více podmínek pro více větví. Vždy bude obsahovat větev `else`, která bude vyhodnocena v případě, že žádná z dříve testovaných podmínek nebude splněna. Pro každou větev bude uživatel muset definovat dva operandy a jeden operátor. Na základě jejich vyhodnocení se poté budou vyhodnocovat bloky odpovídající dané větvi podmínky.

3.2.1.5 Cyklus

Cyklus bude obsahovat jednu řídící podmínku a dále seznam bloků, které se budou vyhodnocovat v případě splnění této podmínky. Cyklus bude postupně procházet všechny záznamy v datech.

3.2.1.6 Přepínač

Přepínač bude odpovídat struktuře `switch` známé z většiny programovacích jazyků. Bude obsahovat řídící proměnnou, která bude porovnávána s jednotlivými hodnotami. Na základě vyhodnocení tohoto porovnání se pak budou vykonávat příslušné bloky.

3.2.1.7 Randomizér

Randomizér bude umožňovat vyhodnocení jen jednoho seznamu bloků. Bude obsahovat více seznamů bloků, ke každému bude možno definovat pravděpodobnost, s jakou se tento seznam bloků vykoná.

3.2.1.8 Odkaz

Odkaz bude umožňovat ukazovat na ostatní bloky v šabloně. Díky němu tak bude možné zabránit nutnosti opakovaného definování různých struktur v šabloně, jelikož bude stačit, když daná struktura bude definována pouze jednou a v případě potřeby dalšího použití se na ní pouze odkáže pomocí odkazu. Na každý blok bude možné odkázat pomocí jeho názvu.

3.3 Automatizace generování zpráv

Pro automatizaci generování textových zpráv bude systém umožňovat definovat jednotlivé zprávy. Tyto zprávy budou obsahovat název zprávy, šablonu, podle které se bude zpráva generovat, seznam všech vygenerovaných zpráv podle tohoto předpisu, seznam uživatelů, kteří budou upozorněni při vygenerování nové zprávy a hlavně čas, ve kterém se bude automaticky spouštět generování dané zprávy.

3.4 Prezentace vygenerovaných zpráv

Vygenerované zprávy budou uživatelům prezentovány dvěma způsoby. Prvním způsobem bude webové rozhraní, ve kterém uživatel uvidí všechny vygenerované zprávy pro danou zprávu. Druhým způsobem bude REST API, které bude po zavolání s identifikačním kódem dané zprávy vracet seznam všech vygenerovaných zpráv.

3.5 Správa uživatelů a skupin

Aby systém reflektoval fungování vydavatelských domů ve světě, bude umožňovat definovat skupiny uživatelů. Skupina tedy bude sdružení uživatelů stejného zájmu –

například právě jeden novinářský dům nebo jedna redakce. V rámci každé skupiny pak bude možné definovat uživatele s různými rolemi – admin, editor a čtenář.

3.5.1 Superadmin

Superadmin bude spravovat skupiny, schvalovat nebo odmítat žádosti o vytvoření nových skupin a bude mít přístup k informacím o všech skupinách. Superadmin nebude součástí žádné skupiny a nebude mít tedy možnost zprávy generovat.

3.5.2 Skupina

Uživatelé budou mít možnost spolupracovat na stejných zprávách a v rámci skupiny je sdílet. Všechny datové objekty budou vázány ke skupině, ke které patří uživatel, jenž daný objekt vytvořil.

3.5.3 Administrátor

Administrátor bude mít oprávnění pro přístup ke všem šablonám, zprávám a skriptům ve své skupině. Bude smět přidávat nové, upravovat a mazat. Také bude mít oprávnění pro vytváření nových uživatelů, změnu jejich rolí a případné smazání uživatelů. Rovněž bude smět skupinu smazat, tím tak smaže všechny objekty na skupinu vázané.

3.5.4 Editor

Editor bude mít, podobně jako v novinářském domě, oprávnění pro vytváření, úpravu a mazání skriptů, šablon a zpráv v dané skupině. Nebude mít žádná oprávnění pro manipulaci s uživateli.

3.5.5 Čtenář

Čtenář bude mít oprávnění pouze pro zobrazení všech zpráv v dané skupině. Bude tak sloužit pouze pro přístup ke zprávám pro uživatele, kteří budou dané zprávy chtít pouze číst.

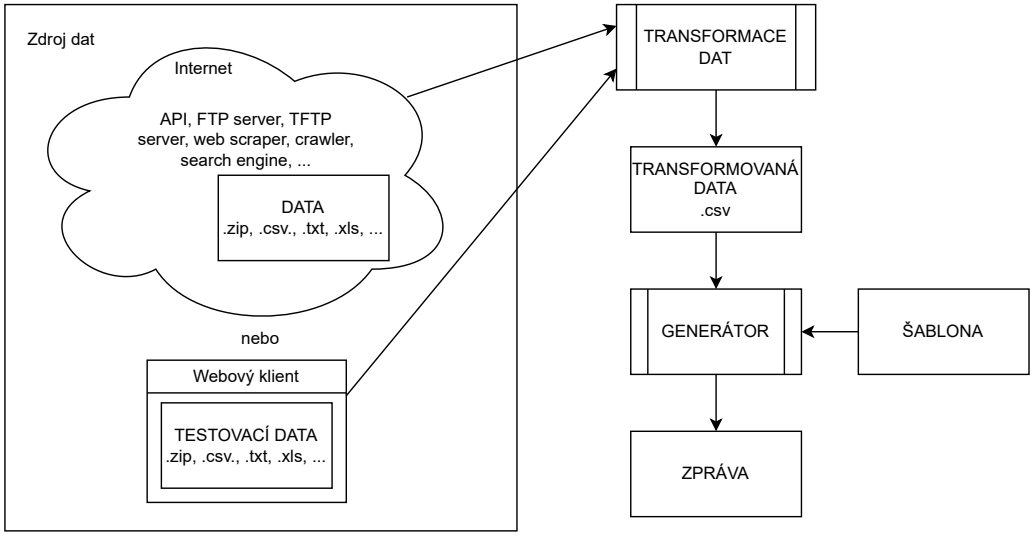
3.6 Softwarová architektura

Systém bude postaven na třívrstvé architektuře MVC, jak je naznačeno na obrázku 3.1 na str. 15. Uživatel bude systém ovládat pomocí webové aplikace. Ta bude pomocí standardního API komunikovat se službou, která bude zpracovávat požadavky, ukládat a načítat data z databáze a odesílat je zpět webovému klientu.

Obrázek 3.1: Architektura systému

3.7 Tok dat

Celkové generování zprávy vyžaduje provedení několika kroků, jež se budou vykonávat sekvenčně. Každý krok vyžaduje určitý vstup a generuje daný výstup. Za provedení každého kroku budou zodpovědné příslušné části aplikace. Tok dat je znázorněn na obrázku 3.2 na str. 15.



Obrázek 3.2: Tok dat

3.7.1 Transformace dat

V prvním kroku generování textu bude z databáze načtena šablona a skript potřebný pro generování dané zprávy. Skript se v prostředí, které bude virtuálně odděleno od zbytku systému, vykoná a jeho výstupem budou soubory s daty, která budou připravena na zpracování v dalším kroku. Virtuální oddělení skriptu od zbytku aplikace bude potřebné pro zajištění bezpečnosti. Bude nutné, aby jednotlivé skripty nemohly zasahovat ať už do serverové aplikace, nebo do běhu ostatních skriptů.

3.7.2 Zpracování dat

Druhým krokem bude předzpracování načtených dat. V tomto kroku bude možné vypočítat průměr z dat, součet dat, najít v datech minimum a maximum, najít v datech první a poslední hodnotu a spočítat počet výskytů určitých hodnot. Výsledky těchto akcí budou uloženy do seznamu konstant. Tento seznam konstant bude uložen dočasně pouze pro potřeby daného generování.

3.7.3 Generování zprávy

Ve třetím kroku bude generována výsledná zpráva podle návodu, který poskytuje šablona, dále podle zpracovaných dat a výsledků akcí pro předzpracování dat. Výsledná zpráva bude poté podle potřeby buď uložena do databáze, nebo vrácena uživateli prostřednictvím standardního API.

3.8 Databáze

Aby bylo možné záznamy o šablonách, zprávách a skriptech opakovaně používat, bude nutné je ukládat do databáze.

3.8.1 Databázové technologie

3.8.1.1 Relační databáze

Relační databáze obvykle používají tabulky s daty uspořádanými do řádků (obsahujících entity) a sloupců (obsahující atributy entit). Tento proces se označuje jako normalizace. Každý řádek obsahuje jedinečný identifikátor nebo klíč, který spojuje tabulky a vytváří vztahy. Při dotazování relační databáze se klíč používá k vyhledání souvisejících dat v rámci datových sad [4]. Mezi nejznámější relační databázové systémy patří MySQL, MariaDB, Oracle či Microsoft SQL.

3.8.1.2 NoSQL databáze

Jedná se o databáze, které nejsou založené na relačním datovém modelu (tj. nejsou relační). Dovedou zpracovávat obrovské množství dat v reálném čase a zpravidla jsou distribuované a otevřené. V databázích se obvykle neřeší transakční zpracování, ale data jsou snadno dostupná vždy i v částečně konzistentním stavu [5]. Dělí se na 4 základní kategorie:

- Klíč-hodnota (např. Redis, Riak) – do databáze se obvykle ukládá dvojice: klíč a jeho hodnota. Na základě znalosti klíče jsme schopni z databáze získat uloženou hodnotu.

- Dokumentové (např. MongoDB, Elasticsearch) – do databáze se ukládají dokumenty ve formátech JSON, případně BSON. Každý ukládaný dokument může mít jinou strukturu.
- Sloupcové (např. Apache Hadoop, Apache Cassandra)- ke každému klíči je možné uložit více hodnot odpovídající příslušnému sloupci. Každý klíč může mít vyplněné hodnoty jiných sloupců.
- Grafové (např. Neo4j, Infinite Graph) – do databáze se ukládají uzly a jejich vlastnosti a také hrany mezi těmito uzly. Hlavním přínosem je vyhledání příslušných uzlů v rozsáhlém grafu na základě implementovaných grafových algoritmů, který je neporovnatelně rychlejší, než běžná relační databáze.

3.8.2 Návrh databáze

Návrh objektů a relací mezi nimi je naznačen na obrázku 3.3 na str. 18. Hlavními objekty ukládanými v databázi budou skupiny (Group, žlutě podbarvená část), uživatelé (User, oranžově podbarvená část), zprávy (Message, modře podbarvená část), šablony (Scheme, zeleně podbarvená část) a skripty (Script, červeně podbarvená část). Šedivě podbarvené objekty pak budou reprezentovat výčtové typy. V databázi budou uloženy také pomocné objekty, které budou pro definici hlavních objektů potřeba. Skupina bude obsahovat reference na přiřazené uživatele, zprávy, šablony a skripty. Zprávy budou odkazovat na vygenerované texty. Uživatelé budou mít referenci na svou skupiny a hodnotu výčtového typu pro určení role daného uživatele. Šablony budou odkazovat na skript vykonávaný před samotným generováním, akce pro předzpracování dat a seznam bloků. Blok bude pouze abstraktní objekt, od něhož budou dědit konkrétní bloky, které budou mít vlastní funkci pro generování textu.

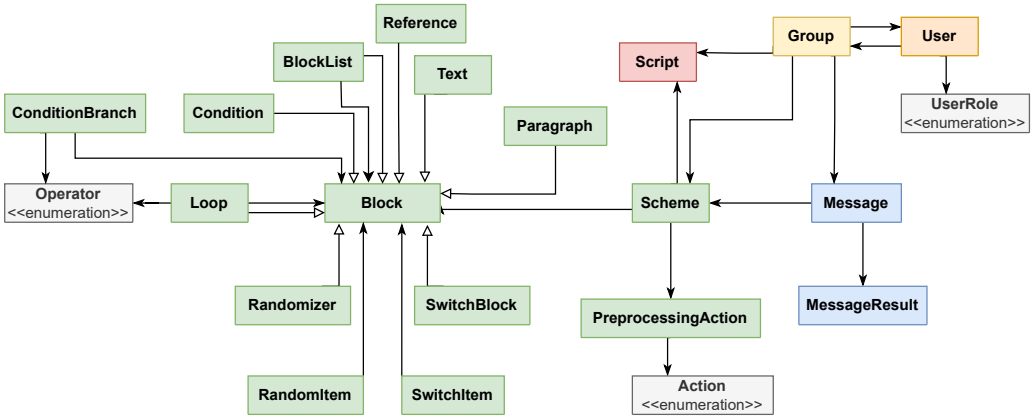
3.9 Webové rozhraní

Webové rozhraní bude sloužit hlavně pro definování šablon. Dále v něm bude možné upravovat skripty, definovat zprávy a spravovat uživatele. Jednoduchý návrh stránek a přechody mezi nimi je zobrazen na obrázku 3.4 na str. 20.

3.9.1 Webové technologie

3.9.1.1 React

React je JavaScriptová knihovna sloužící k tvorbě webového rozhraní. Princtip reakcu je založen na komponentách, kdy každá komponenta má své proměnné, metody a HTML šablonu pro vykreslení popisované komponenty. Komponenty jsou



Obrázek 3.3: Návrh databázových objektů

následně do sebe skládány tak, aby vytvořily kompletní webovou stránku. React poskytuje pouze základní funkce pro tvorbu webových stránek, pro pokročilé funkce je nutné React doplnit o další knihovny [6]. React byl od roku 2011 vyvíjen společností Facebook, v roce 2013 byl pak publikován pro veřejnost [7].

3.9.1.2 Angular

Angular je JavaScriptový framework, který na rozdíl od Reactu poskytuje všechny nástroje potřebné pro vývoj webových stránek. Podobně jako React se výsledné webové stránky skládají z komponent. Každá komponenta má také své proměnné, metody a HTML šablonu pro popis svého vzhledu [8]. Pro programování v první verzi Angularu byl použit JavaScript. Od jeho dalších verzí už je ale používán TypeScript. TypeScript je nadstavba nad JavaScriptem a doplňuje jej o statické typování, třídy a další vlastnosti převzaté z OOP. Pro spuštění TypeScriptu je nutné použít transpiler. To je druh kompilátoru, který ze zdrojového kódu psaného v TypeScriptu vygeneruje stejně fungující zdrojový kód v JavaScriptu. Angular je od roku 2010 vyvíjen společností Google [9].

3.9.2 Návrh stránek

3.9.2.1 Přihlášení

Tato stránka bude sloužit pro přihlášení uživatele, bude odkazovat na registraci.

3.9.2.2 Domovská stránka

Na domovské stránce budou základní informace o přihlášeném uživateli a skupině, ke které je přiřazen, přehled posledních generovaných zpráv a seznam zpráv, které

se budou v nejbližší době generovat.

3.9.2.3 Administrace skupin

Tato stránka bude přístupná pouze uživateli s rolí Superadmin. Zde bude přehled všech skupin zaregistrovaných do systému a přehled skupin, které čekají na schválení. O každé skupině uvidí uživatel základní informace jako jsou kontaktní údaje, počet zpráv, uživatelů atd.

3.9.2.4 Administrace skupiny

Administrace skupiny bude přístupná pouze uživateli s rolí Admin. Na této stránce bude možné upravovat role ostatních uživatelů, přidávat nové uživatele, mazat uživatele a smazat skupinu.

3.9.2.5 Profil uživatele

Na této stránce bude mít každý přihlášený uživatel možnost zobrazit své údaje, případně je změnit a také zde bude mít možnost si změnit heslo.

3.9.2.6 Přehled šablon

Zde bude tabulka se všemi šablonami, které byly vytvořeny. Uživatel s rolí Admin nebo Editor bude mít možnost otevřít, smazat, upravit, nebo vytvořit novou šablonu.

3.9.2.7 Tvorba a úprava šablon

Tato stránka bude sloužit pro tvorbu nových šablon či úpravu šablon již existujících. Uživatel s rolí Admin nebo Editor zde bude mít možnost definovat akce pro předzpracování dat, vybrat skript, který se bude před vyhodnocováním šablony vykonávat, a hlavně sestavit samotnou šablonu pro výslednou zprávy.

3.9.2.8 Přehled skriptů

Bude obsahovat tabulku se všemi dostupnými skripty. Uživatel s rolí Admin nebo Editor bude mít možnost otevřít, upravit, smazat nebo vytvořit nový skript.

3.9.2.9 Tvorba a úprava skriptů

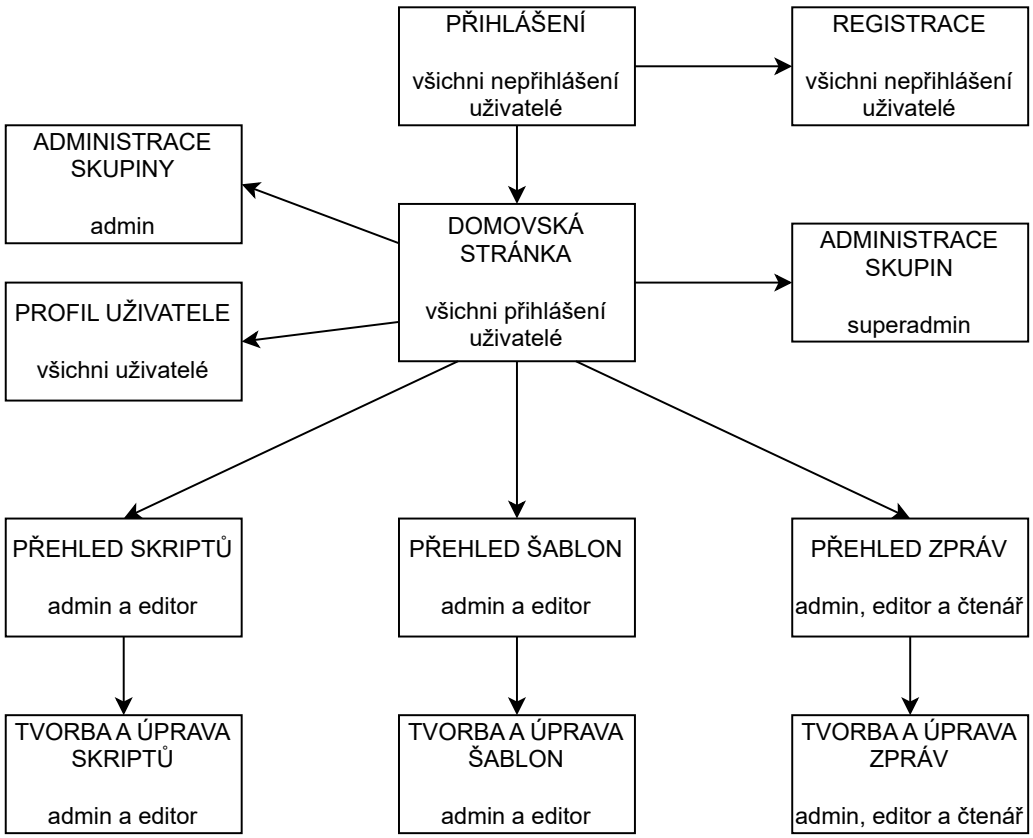
Tato stránka bude sloužit pro tvorbu nových či úpravu stávajících skriptů. Uživatel s rolí Admin nebo Editor zde bude mít možnost nahrát soubor s daným skriptem, upravit jej v editoru a definovat příkaz, kterým se bude skript spouštět.

3.9.2.10 Přehled zpráv

Zde se bude nacházet tabulka se všemi dostupnými definicemi zpráv. Všichni přihlášení uživatelé budou mít povolení k otevření nějaké zprávy. Pouze uživatelé s rolí Admin nebo Editor budou mít povolení k otevření zprávy pro její úpravu.

3.9.2.11 Tvorba a úprava zpráv

Na této stránce bude pro uživatele s rolí Admin nebo Editor možné upravovat již existující definici zprávy nebo vytvořit novou. Bude zde možné definovat název zprávy a šablonu použitou pro její generování. Všichni přihlášení uživatelé zde také uvidí všechny zprávy vygenerované podle této definice.



Obrázek 3.4: Schéma stránek

3.10 Server

Server bude poskytovat sadu API, pomocí kterých bude možné ukládat, vybírat a upravovat data z databáze.

3.10.1 Serverové technologie

3.10.1.1 Express

Express je JavaScriptový framework pro vývoj aplikací poskytující webové API. Pro běh Expressu se běžně využívá serverové prostředí Node.js, které kód napsaný v tomto frameworku spouští [10]. Express umožňuje jednoduchou definici REST API.

3.10.1.2 SpringBoot

SpringBoot je framework pro vývoj aplikací poskytující webové API psaný v programovacím jazyce Java. SpringBoot má v sobě zabudovaný servlet kontejner, díky němuž je při spuštění aplikace automaticky nastartován server naslouchající příchozím požadavkům pomocí HTTP [11]. Pomocí rozhraní definovaných SpringBootem je možné jednoduše přistupovat k databázi.

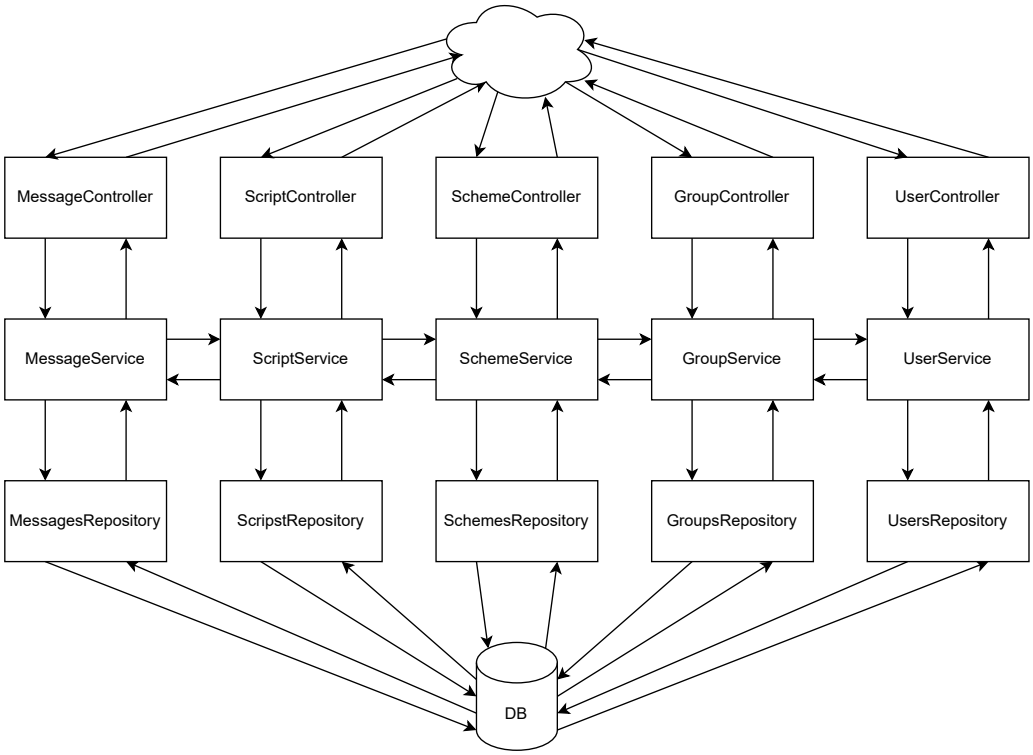
3.10.2 Architektura

Server bude implementován třívrstvou architekturou, jako je naznačeno na obrázku 3.5 na str. 22, kdy každý z požadavků obsahující nějaký z hlavních objektů je obsloužen v příslušném kontroléru (např. zprávy budou obslouženy kontrolérem `MessageController`). Kontrolér pouze definuje rozhraní (API) a obsluhuje dané požadavky pomocí příslušné služby (např. `MessageController` komunikuje s `MessageService`). Služba podle zadaných parametrů provede daný úkol a s databází komunikuje pomocí repozitářů (např. `MessageService` komunikuje s `MessageRepository`). Repositář provádí pouze jednoduché operace nad databází, např. maže objekty, ukládá nové objekty, upravuje objekty a vyhledává objekty podle zadaných parametrů. Každá vrstva po vykonání svého úkolu vrací vyšší vrstvě výsledek své práce a nakonec kontrolér odpovídá na daný požadavek.

3.10.3 Přístupové body rozhraní (API)

Pro každý hlavní databázový objekt (zpráva, šablona, skript, uživatel, skupina) budou implementovány 4 základní API pro operace s těmito objekty:

- `create` – uloží daný objekt do databáze,
- `read` – přečte daný objekt z databáze,
- `update` – upraví objekt uložený v databázi,
- `delete` – smaže daný objekt z databáze.



Obrázek 3.5: Architektura serveru

3.10.3.1 Zprávy

Rozhraní pro databázový objekt „Zpráva“ bude kromě základních API poskytovat ještě přístupové body pro získání hlaviček všech zpráv, duplikování dané zprávy, ruční spuštění generování textu podle zadané zprávy, mazání vygenerovaných textů dané zprávy a generování textu z testovacích souborů.

3.10.3.2 Šablony

Základní rozhraní bude pro databázový objekt „Šablona“ doplněno o API poskytující všechny operátory a akce pro předzpracování dat použitelné v šabloně, duplikování dané šablony, načtení všech šablon a převedení šablony z formátu používaného webovou aplikací do databázové podoby.

3.10.3.3 Skripty

Rozhraní pro databázový objekt „Skript“ bude základní API rozšiřovat pouze o rozhraní pro duplikování daného skriptu a načtení hlaviček všech skriptů.

3.10.3.4 Uživatelé

Oproti základnímu API bude rozhraní pro databázový objekt „Uživatel“ poskytovat přístupové body pro získání použitelných uživatelských rolí, přihlášení uživatele, ověření platnosti přihlášeného uživatele a změnu role uživatele.

3.10.3.5 Skupiny

Základní API bude pro databázový objekt „Skupina“ rozšířeno pouze o rozhraní pro schválení žádosti o vytvoření skupiny a rozhraní pro načtení hlaviček všech skupin uložených v databázi.

Implementace 4

4.1 Databáze

4.1.1 Technologie

Jako databázi pro uchování datových struktur jsem vybral NoSQL databázi MongoDB. Kategorii NoSQL databází jsem zvolil z důvodu jejich pružnosti, rychlosti, a dřívějšího seznámení se s nimi. Konkrétně MongoDB jsem pak zvolil z důvodu dobré podpory frameworku, pomocí kterého jsem se rozhodl implementovat serverovou část. MongoDB je dokumentová databáze, a tak dovoluje ukládat objekty z jazyka Java jako JSON struktury. Tyto struktury navíc nemusejí být úplné a databáze se dobře vypořádává s úpravami (ve smyslu změny implementace) tříd, které tyto objekty definují.

4.1.2 Databázový model

Na obrázku 4.1 na str. 38 je vyobrazen celý databázový model. Databáze se bude skládat z pěti hlavních dokumentů – `Message` (reprezentující zprávy, modře podbarvené tabulky), `Scheme` (reprezentující šablony, zeleně podbarvené tabulky), `Script` (reprezentující skripty, červeně podbarvená tabulka), `User` (reprezentující uživatele, oranžově podbarvená tabulka) a `Group` (reprezentující skupiny, žlutě podbarvená tabulka). Šedivě podbarvené tabulky pak reprezentují výčtové typy. Tyto dokumenty budou ukládány do příslušných kolekcí (např. dokumenty typu `Group` budou uloženy v kolekci `groups`). Každý dokument, jenž je uložen v kolekci, musí být jednoznačně identifikován. K tomu budou využita pole `id` (kromě dokumentu `User`, ten bude identifikován polem `login`), která jsou na obrázku 4.1 na str. 38 znázorněna tučným textem. Každý dokument je popsán sadou atributů. Tyto atributy musí korespondovat s příslušnou implementací daného dokumentu v jazyce Java. Dokumenty jsou implementovány v balíku `com.julrych.data2text`. Vztahy mezi jednotlivými dokumenty jsou naznačeny plnými šipkami. Prázdnými šipkami je poté naznačeno dědění jednotlivých tříd v Javě.

4.2 Webová aplikace

4.2.1 Technologie

Pro implementaci webové části jsem zvolil framework Angular, který slouží pro tvorbu webových aplikací. Konkrétně jeho verzi 13, která jako technologie pro popis webových stránek používá HTML, CSS, a TypeSript. Tyto technologie jsem zvolil z důvodu svých předchozích zkušeností s nimi.

4.2.2 Architektura

Framework Angular je založen na komponentách. Komponenta je část webové stránky, která se dá například opakovaně používat na více místech. Příkladem takové komponenty může být navigační panel. Komponentou může být ale i celá stránka, což je případ, který byl při implementaci volen pro většinu příležitostí, jelikož komponenty nebylo potřeba používat na více místech v aplikaci.

4.2.3 Struktura projektu

4.2.3.1 Kořenový adresář

V kořenovém adresáři webové aplikace se nachází soubory potřebné pro kompilaci a sestavení aplikace frameworkem Angular, konfigurační soubory a návod pro sestavení pro použití v rámci Dockeru. Dále se v něm nachází adresář `src`, jenž obsahuje zdrojové soubory aplikace.

4.2.3.2 Adresář `src`

V tomto adresáři se nacházejí všechny zdrojové soubory a ostatní pomocné soubory, které jsou potřeba k úplnému vykreslení webových stránek. Všechny soubory jsou roztříděny do souborů podle své kategorie.

4.2.3.3 Adresář `src/app`

Tento adresář obsahuje pouze zdrojové soubory potřebné k běhu aplikace. Obsahuje komponentu `AppComponent`, jež je kořenovou komponentou a nad níž se vykreslují všechny ostatní komponenty podle příslušné URL adresy. Jednotlivé podadresáře jsou popsány v následujících odstavcích.

4.2.3.4 Adresář src/app/layouts

Definuje komponentu `AdminLayoutComponent`, jež má na starost správné vykreslení navigačního panelu stránky, vlastního obsahu stránky (závislý na dané URL adrese) a patičky stránky,

4.2.3.5 Adresář src/app/pages

Obsahuje jednotlivé komponenty aplikace. Každá komponenta odpovídá jedné stránce aplikace, například komponenta `SchemeEditComponent` odpovídá stránce pro tvorbu, úpravu a testování jednotlivých šablon.

4.2.3.6 Adresář src/app/services

Obsahuje služby, které poskytují komunikační prostředek mezi webovou aplikací a serverovou částí aplikace. Pro každý kontrolér v serverové části je zde jedna služba, kterou je možno volat z jakékoli komponenty a díky které se tak uživatelské akce mohou propsat do aplikace. Například po kliknutí na tlačítko „uložit“ na stránce s šablonou zavolá metoda `saveScheme()` z komponenty `SchemeEditComponent` metodu `saveScheme(SchemeInterface)` služby `SchemeService`, která poté provede HTTP požadavek na URL adresu daného API pro uložení šablony. Odpověď na tento požadavek poté předá volající komponentě, která jej zpracuje a zobrazí uživateli informaci o uložení šablony. Jediná služba, která neodpovídá kontroléru v serverové části, je služba `AuthGuardService`. Ta nevykonává žádné požadavky vůči serveru, ale pouze kontroluje, zda má právě přihlášený uživatel právo přistoupit k dané stránce.

4.2.3.7 Adresář src/app/shared

Definuje navigační panel a patičku aplikace. Z navigačního panelu je možné se přepínat mezi jednotlivými stránkami, přihlašovat se a zobrazovat uživatelský profil. Patička aplikace obsahuje odkaz na stránku s tutoriály.

4.2.3.8 Adresář src/assets

Obsahuje podadresáře se soubory, které jsou v aplikaci na různých místech potřeba a nedají se označit jako zdrojový kód.

- Podadresář `fonts` – obsahuje soubory pro výchozí font textů v aplikaci a také font používaný k vykreslení ikon.
- Podadresář `img` – obsahuje obrázky používané v různých částech aplikace, primárně však obrázky používané na stránce s tutoriály.

- Podadresář `scripts` – obsahuje ukázkové skripty, které si může uživatel stáhnout ze stránky s tutoriály.
- Podadresář `scss` – obsahuje soubory s kaskádovými styly pro všechny ovládací prvky aplikace.
- Podadresář `schemes` – obsahuje ukázkové šablony, které si může uživatel stáhnout ze stránky s tutoriály.

4.2.3.9 Adresář `src/data-types`

Obsahuje rozhraní (ve smyslu `interface` z jazyka TypeScript), která svými atributy odpovídají databázovému modelu. Tato rozhraní jsou poté v aplikaci využívána k přenosu dat mezi jednotlivými komponentami a službami. Data ze serveru jsou ve valné většině automaticky namapována na tato rozhraní. Jedinou výjimkou jsou rozhraní pro jednotlivé bloky šablony, ty musí být kvůli jejich odlišnosti od databázového modelu mapovány manuálně pomocí metod ze třídy `utils.MapUtilsClass`.

4.2.3.10 Adresář `src/environments`

Tento adresář obsahuje konfigurační soubory pro různé prostředí běhu aplikace, konkrétně pro vývojové (soubor `environment.ts`) a produkční (soubor `environment.prod.ts`) prostředí.

4.2.3.11 Adresář `src/utils`

Adresář `src/utils` obsahuje pomocné třídy, jejichž metody jsou volány z různých míst v kódu komponent. Třída `ConstantUtil` obsahuje deklarace statických konstant. Třída `GeneralUtils` obsahuje například metodu pro generování ID pro odlišení textových polí pro vstup od uživatele. Třída `MapUtils` poskytuje metody pro mapování objektů z databáze na objekty používané webovou aplikací.

4.3 Serverová aplikace

4.3.1 Technologie

Jako programovací jazyk pro serverovou část jsem zvolil Javu ve verzi 17. Jako framework, který mi pomůže s implementací REST API a komunikací s databází, jsem zvolil SpringBoot. Ten má v sobě zabudovaný server, díky němuž se budou moci uživatelé připojovat k rozhraní pomocí HTTP požadavků. V rozhodování nad technologiemi mi pomohla i má předchozí znalost těchto technologií.

4.3.2 Skript pro zpracování dat

Jazyk pro skript, který bude načítat data z různých zdrojů a transformovat je do dále zpracovatelné podoby jsem zvolil skriptovací jazyk Python. Důvodem zvolení právě Pythonu je jeho dostatečná modularita, jednoduchost, relativně široká znalost a velké množství podpůrných knihoven pro zpracování dat.

4.3.3 Data zpracovávána šablonou

Formát dat zpracovávaných šablonou (tedy těch, které by měly být výstupem skriptu) jsem zvolil CSV, což je jednoduchý souborový formát určený pro uchování tabulkových dat. Soubor by měl v první řádce obsahovat názvy všech sloupců oddělené středníkem. Na dalších řádcích jsou pak hodnoty daných záznamů pro příslušné sloupce, opět odděleny středníkem.

4.3.4 Implementace API

Jednotlivá API jsou definována ve třídách (kontrolérech) nacházejících se v balíku `com.julrych.data2text.api`. API jsou rozdělena tématicky do jednotlivých kontrolérů dle databázových objektů, se kterými pracují, nebo ke kterým se váží. Například kontrolér `ScriptController` obsahuje 4 základní rozhraní – `/getScripts` (vrátí hlavičky všech skriptů), `/getScriptDetail` (vrátí detail jednoho skriptu), `/saveScript` (uloží skript) a `/deleteScript` (smaže skript). Obdobně je tomu tak pro všechny ostatní kontroléry.

4.3.5 Základní odpověď na požadavky

Odpověď na volání API se pro každé API liší. Všechny odpovědi ale mají společná tři pole, jež jsou zobrazena v kódu 4.1 na str. 29. Pole `errorMessage` obsahuje chybovou zprávu v případě chybného zpracování požadavku. Pole `stackTrace` pak obsahuje výpis zásobníku v případě vyhození výjimky v průběhu zpracování požadavku. Pole `responseCode` obsahuje číslo reprezentující stav vyhodnocení požadavku. V případě správného vyhodnocení je `responseCode` nastaven na hodnotu `200`, pole `errorMessage` a `stackTrace` jsou pak nastavena na hodnotu `null`. Na základě typu požadavku pak odpověď obsahuje další pole. Konkrétní odpovědi na dané požadavky jsou detailně popsány v dokumentaci.

Zdrojový kód 4.1: Základní odpověď na požadavky

```
{
    "errorMessage": null,
    "responseCode": 200,
    "stackTrace": null
}
```

4.3.6 Architektura

Serverová část byla implementována podle navržené třívrstvé architektury. První vrstva kontrolérů přijímá požadavky a komunikuje s vrstvou služeb, které na základě parametrů tyto požadavky zpracovávají a komunikují s vrstvou repositářů, které provádějí triviální operace nad databází.

4.3.6.1 Kontroléry

Kontroléry jsou implementovány v balíku `com.julrych.data2text.api`. Aby mohly požadavky přijímat musí se dle pravidel frameworku SpringBoot označit anotací `org.springframework.web.bind.annotation.RestController`. Pro jednotlivá API pak musí být definována metoda HTTP požadavku (GET, POST, PUT, ...) a také adresa, která bude pro daný požadavek použita. Toto je opět definováno pouze pomocí anotací (např. `@GetMapping("getScripts")`).

4.3.6.2 Servisní vrstva

Služby jsou implementovány v balíku `com.julrych.data2text.service`. SpringBoot využívá návrhový vzor Dependency Injection, díky kterému se vývojář nemusí starat o vytváření instancí služeb, ale framework je vytvoří automaticky za něj. Aby toto mohlo fungovat, musí se služby označit anotací `org.springframework.stereotype.Service`. Jednotlivé služby definují veřejné metody, které jsou využívány kontroléry pro odbavování požadavků a privátní metody určené pro interní zpracování požadavků.

4.3.6.3 Repositáře

Repositáře jsou implementovány v balíku `com.julrych.data2text.repositories`. Repositář je rozhraní, které je označeno anotací `org.springframework.stereotype.Repository` a implementuje rozhraní `org.springframework.data.mongodb.repository.MongoRepository` s dvěma typovými parametry – třídou definující daný databázový objekt a datovým typem identifikačního pole tohoto databázového objektu. Framework SpringBoot nabízí v repositářích ve výchozím stavu metody pro ukládání dokumentů do databáze, smazání podle ID, vyhledání podle ID a výběr všech dokumentů. Založení dalších dotazů do databáze pak spočívá pouze v definici daného dotazu jakožto metody daného rozhraní podle předem daných pravidel.

4.3.6.4 Implementace databázového modelu

Všechny třídy, jejichž objekty jsou ukládány do databáze jakožto dokumenty, jsou implementovány v příslušných podbalících balíku `com.julrych.data2text.model`. Z pohledu návrhových modelů se jedná o jednoduché přepravky. Atributy těchto tříd musí přesně odpovídat databázovému modelu definovanému výše. V případě, že nechceme nějaká data ukládat do databáze, použijeme u definice tohoto atributu anotaci `org.springframework.data.annotation.Transient`. V opačném případě, kdy chceme nějaká data ukládat do databáze, ale nechceme je poskytovat uživateli, použijeme u příslušného atributu anotaci `com.fasterxml.jackson.annotation.JsonIgnore`.

4.3.7 Struktura projektu

4.3.7.1 Kořenový adresář

V kořenovém adresáři se nachází adresář `src` se zdrojovými kódy projektu, adresář se soubory potřebnými pro spuštění uživatelsky definovaného skriptu v prostředí Dockeru a další konfigurační soubory potřebné pro spuštění projektu, jako je například soubor `pom.xml`, který popisuje projekt z pohledu balíkovacího systému Maven, nebo soubor Dockerfile, který popisuje sestavení projektu pro spuštění v Dockeru.

4.3.7.2 Adresář `docker-script-data`

Tento adresář obsahuje soubor Dockerfile, jenž je použit při spuštění generování textu pro oddělení uživatelsky definovaného skriptu od zbytku aplikace. Každý skript je spuštěn v samostatném Docker kontejneru z důvodu ochrany dat samotné serverové aplikace. Díky spuštění v Docker kontejneru je právě spuštění skript naprosto oddělen od dat serveru. Výměna potřebných souborů se serverem je realizována přes složku `output_files` v kontejneru daného skriptu, která je mapována do složky `/tmp/script/<id>`, jenž se nachází v systému, kde běží serverová aplikace. Dalším souborem v tomto adresáři je soubor `requirements.txt`, který obsahuje seznam závislostí pro Python. Do tohoto souboru je možné případně přidat další požadované závislosti.

4.3.7.3 Adresář `src`

Adresář `src` obsahuje v podadresáři `main/java` zdrojové kódy aplikace rozřazené do balíků. Dále budou popsány jednotlivé balíky projektu. Názvy všech balíků začínají předponou `com.julrych.data2text`.

4.3.7.4 Balík com.julrych.data2text.api

Tento balík obsahuje třídy `GroupController`, `MessageController`, `SchemeController`, `ScriptController`, `UserController` a `AbstractController`, jenž implementují API popsané v 3.10.3 na str. 21. Od poslední jmenované třídy `AbstractController` všechny kontroléry dědí. Tato třída poskytuje metodu pro řešení výjimek. Výjimky jsou kontroléry zpracovávány tak, že jejich stručný i rozsáhlý popis je nastaven do odpovědi na požadavek. Výjimka je také vypsána do logu aplikace.

4.3.7.5 Balík com.julrych.data2text.config

Tento balík obsahuje programovou konfiguraci serveru. Třída `AppConfig` po nastartování aplikace ukládá uživatele s rolí Superadmin do databáze.

4.3.7.6 Balík com.julrych.data2text.data

V balíku `com.julrych.data2text.data` jsou definovány třídy používané při generování textu. Třída `CsvData` slouží jako přepravka pro data načtená z `.csv` souboru a poskytuje k těmto datům ucelený přístup. Třída `GenerationEnvironment` udržuje stav všech proměnných konkrétního generování textu. Konkrétně svými atributy drží všechna načtená data, výsledky akcí pro předzpracování dat a aktuální index cyklu. Třída `TextGenerationResult` pak slouží jako přepravka pro výsledky konkrétního generování textu, drží v sobě vygenerovaný text, soubor s logy v textovém řetězci ve formátu Base64 a případně vyhozenou výjimku.

4.3.7.7 Balík com.julrych.data2text.enums

Tento balík obsahuje různé výčtové typy použité v aplikaci. Hodnoty výčtového typu `Action` reprezentují všechny možné akce, které může uživatel definovat v rámci předzpracování dat. Výčtový typ `Operators` pak svými hodnotami reprezentuje možné operátory použitelné pro definování podmínek v šabloně. Nakonec hodnoty výčtového typu `UserRole` reprezentují všechny možné role, které mohou uživatelé používat. Každá uživatelská role má navíc přiděleno celé číslo, které udává její důležitost oproti ostatním.

4.3.7.8 Balík com.julrych.data2text.exceptions

V balíku `com.julrych.data2text.exceptions` se nacházejí všechny výjimky specifické pro tuto aplikaci. Žádná metoda, která vyhazuje výjimku, by neměla vyhazovat generickou výjimku `java.lang.Exception`, ale pouze tu, která je definována v tomto balíku, či jinou, která od `java.lang.Exception` dědí.

4.3.7.9 Balík com.julrych.data2text.model

V tomto balíku je implementován databázový model pomocí objektů s atributy, které přesně odpovídají atributům jednotlivých databázových objektů. Většina z těchto objektů má implementován konstruktor s jedním parametrem typu `com.fasterxml.jackson.databing.JsonNode`, což je objekt, který odpovídá jednomu uzlu ve formátu JSON. Tento konstruktor se z daného JSON pokusí získat potřebné atributy pro vytvoření instance daného objektu, a pokud se to podaří, volá příslušný parametrický konstruktor dané třídy. O logiku načítání dat z JSON formátu se tak stará každá třída samostatně. Třída `AbstractDocument` pak slouží jako předek těm objektům, které musí být v databázi identifikovány jednoznačným řetězcem, neboť poskytuje metody pro vytvoření tohoto jednoznačného identifikátoru. Každý hlavní databázový objekt (tedy ten, pro který je v databázi vytvořena samostatná kolekce a je pro něj potřeba implementovat repositář, musí být označen anotací `org.springframework.data.mongodb.core.mapping.Document` s parametrem, který udává název kolekce daných objektů). Pokud chceme jako atribut jednoho hlavního databázového objektu uvést další databázový objekt, je nutné tento atribut označit anotací `org.springframework.data.mongodb.core.mapping.DBRef`.

4.3.7.10 Balík com.julrych.data2text.repositories

V balíku `com.julrych.data2text.repositories` jsou implementovány repositáře pro jednotlivé databázové objekty. Každý repositář musí dědit od rozhraní `org.springframework.data.mongodb.repository.MongoRepository` a uvést dva typové parametry – prvním z nich je datový typ databázového objektu a druhým pak datový typ pole, které daný objekt jednoznačně identifikuje. Repositář MongoRepository poskytuje základní operace pro vyhledání, uložení a mazání objektů, všechny tyto operace se váží k ID daného objektu. Implementace operací, které se váží k jiným atributům daného objektu, je díky SpringBoot frameworku snadná. Stačí v rozhraní pouze deklarovat metodu s názvem, který odpovídá předepsanému standardu frameworku SpringBoot a ten se již poté automaticky postará o správnou exekuci těchto operací.

4.3.7.11 Balík com.julrych.data2text.response

V tomto balíku se nachází třídy sloužící jako odpovědi na jednotlivé požadavky. Tyto třídy jsou v podstatě pouze jednoduché přepravky, ale dobře pomáhají při přenosu dat ve formátu JSON. Každá třída poskytující atributy, které nesou konkrétní odpověď na daný požadavek, musí dědit od třídy `AbstractResponse`. Ta poskytuje

přístupové metody pro atributy, které musí mít každá odpověď tak, jak je popsáno v 4.3.5 na str. 29.

4.3.7.12 Balík `com.julrych.data2text.service`

Balík `com.julrych.data2text.service` obsahuje implementace všech služeb pro manipulaci s databázovými objekty. Každá třída odpovídá jednomu kontroléru z balíku `com.julrych.data2text.api` a poskytuje veřejnou metodu pro každou metodu daného kontroléru. Služba by měla podle předaných parametrů z kontroléru provést požadovanou operaci nad jedním či více databázovými objekty a v návratové hodnotě vrátit příslušnou hodnotu. Pro přístup k databázi využívá jednotlivých repositářů. Pokud se operaci z jakéhokoli důvodu nepodaří vykonat, vyhodí daná metoda výjimku odpovídající dané chybě při zpracování požadavku. Tato výjimka je následně zpracována volajícím kontrolérem. Služba `ScheduleTaskService` nemá odpovídající kontrolér, slouží pouze pro správu událostí, které se musí vykonat v čase jiném, než aktuálním. Primárně slouží pro automatické spouštění generování textu v uživatelsky definovaný čas. Služba `MailService` taktéž nemá svůj protějšek mezi kontroléry, slouží pouze pro odesílání emailů.

4.3.7.13 Balík `com.julrych.data2text.utils`

V tomto balíku se nachází pomocné třídy. Třída `ExceptionProvider` poskytuje metody pro jednoduchou tvorbu výjimek. Třída `JsonParser` pak poskytuje metody pro jednoduchou extrakci dat z `com.fasterxml.jackson.databind.JsonNode` objektů.

4.3.8 Proces generování textu z pohledu serveru

Na obrázku 3.2 na str. 15 je znázorněn celkový tok dat při procesu generování textu. Níže bude popsáno generování textu z pohledu serveru – volané metody, požadované parametry a výstupy v jednotlivých krocích.

4.3.8.1 Spuštění generování textu

Generování textu může být spuštěno několika způsoby:

- manuálně pomocí API `/generateText` – využíváno zejména pro otestování implementované šablony uživatelem,
- manuálně pomocí API `/runMessage` – využíváno pro manuální spuštění generování textu z uložené definice zprávy,

- automaticky pomocí předem definovaného času pro spuštění generování textu pro příslušnou zprávu.

Všechny tyto způsoby mají společné to, že musí volat metodu `generateText(List<Block>, ...)` třídy `MessageService`, která je vstupním bodem pro každé generování textu. Vstupními parametry této metody je seznam bloků použitých pro generování (tedy v podstatě šablona), seznam akcí pro předzpracování dat, ID skriptu, testovací data a parametry skriptu. Poslední dva jmenované parametry jsou ve formátu textového řetězce a mohou být prázdné.

4.3.8.2 Dočasné prostředí pro generování

Každé generování textu je označeno jednoznačným textovým identifikátorem. V adresáři `/tmp/scripts` je před začátkem generování textu vytvořen podadresář s názvem `script-exec-<ID>`, do kterého budou ukládány soubory potřebné pro dané generování a výstupy z něj. Tento adresář není ihned po dokončení generování smazán, ale ponechává se pro účely zpětné kontroly a nalezení případných chyb při generování.

4.3.8.3 Vykonání skriptu

O vykonání skriptu se stará soukromá metoda `executeScript` třídy `MessageService`. Tato metoda nejprve vytvoří složku popsanou v odstavci výše, nakopíruje do ní případný soubor s daty (pouze pokud je předán), vytvoří v ní soubor se skriptem, vytvoří soubor Dockerfile s parametry odpovídajícími danému skriptu a nakopíruje do složky soubor `requirements.txt` s požadavky na knihovny pro Python. Poté vytvoří nový proces, který nejprve vytvoří obraz z daného Dockerfile, následně vytvoří obraz disku, který propojuje serverové prostředí s prostředím kontejneru (serverový adresář `/tmp/scripts/script-exec-<ID>` na kontejnerový adresář `/output_files`, ve kterém bude spuštěn skript, následně spustí kontejner s uživatelským skriptem a po skončení tohoto kontejneru smaže jeho obraz. Pokud je v konfiguraci aplikace povoleno použití sdíleného prostoru pro odkládání dat, společně s propojením serverového prostředí s kontejnerem se provede navíc propojení serverového adresáře `/tmp/tmp` s kontejnerovým adresářem `/tmp`. Soubory, které tedy skript z nějakého zdroje získá, je možno do adresáře `/tmp` uložit a použít při budoucím generování textu. Soubory, které uživatelský skript vygeneruje, musí být ukládány do adresáře `output_files`. Z tohoto adresáře metoda následně vybere všechny soubory, které začínají předponou `output`, tyto soubory zakóduje do formátu Base64 a vrátí.

4.3.8.4 Načtení dat ze souborů

Soubory kódované ve formátu Base64 jsou následně v metodě `generateText` předány konstruktoru třídy `CsvData`, který tyto soubory načte, pro každou hlavičku načte její data a tato data si uloží do svých proměnných. Výsledkem je tak seznam instancí tříd `CsvData`, kdy každá jedna třída reprezentuje jeden soubor vygenerovaný uživatelským skriptem.

4.3.8.5 Předzpracování dat

Metoda po načtení dat vyhodnotí akce pro předzpracování dat definované uživatelem a jejich výsledky uloží do mapy, kde klíč odpovídá hodnotě, kterou uživatel zadal do sloupce „výsledek“ při definování těchto akcí a hodnota spjatá s tímto klíčem je pak výsledkem této akce.

4.3.8.6 Nahrazení odkazů

Všechny bloky typu „odkaz“ jsou následně nahrazeny za bloky, na které odkazovaly. Rekurzivně je prohledáván seznam všech bloků až do chvíle, dokud není nalezen blok s daným názvem či ID. Po nalezení tohoto bloku je tento blok nakopírován na místo, kde je odkazující blok. Pokud blok s daným názvem či ID není nalezen, odkazující blok je pouze smazán a do logů serveru je vypsána informace o této skutečnosti. To, jestli odkaz odkazuje na validní blok, kontroluje klient pomocí vizuálních nápověd uživateli. Pokud je při rekurzivním hledání odkazovaného bloku překročena určitá hloubka zanoření (momentálně nastavena na hodnotu 100), je vyhozena výjimka.

4.3.8.7 Vyhodnocení bloků

Po nahrazení odkazů je přistoupení k vlastnímu vyhodnocení všech bloků definovaných v šabloně. Metoda `evaluateBlocks` rekurzivně prochází seznam všech bloků, každý z nich vyhodnocuje, a pokud se jedná o textový blok, připojí k již vygenerovanému textu příslušný text. Hloubka zanoření rekurze je zde opět limitována na hodnotu 100, při jejím překročení je vyhozena výjimka. Při vyhodnocování bloků je potřeba znalost dat (tzn. všechny instance tříd `CsvData` držící načtená data), znalost aktuální pozice v šabloně (např. aktuální index v cyklu), výsledky akcí pro předzpracování dat a dosavadně vygenerovaný text.

4.3.8.8 Výstup generování

Výstupem metody je instance třídy `TextGenerationResult`, ta obsahuje vygenerovaný text, v případě vyhození výjimky i tuto výjimku a pro potřeby analýzy

generování také archivní soubor s logy generování. Soubor s logy je uložen jako textový řetězec v kódování Base64 a obsahuje následující soubory:

- `stdout.log` – standardní výstup uživatelsky definovaného skriptu,
- `stderr.log` – chybový výstup uživatelsky definovaného skriptu,
- `execution_times.log` – obsahuje celkovou délku generování textu, délku vykonávání skriptu a délku vyhodnocování jednotlivých bloků,
- `java_exception.log` (pouze při vyhození výjimky) – detailní výpis obsahu vyhozené výjimky,
- soubory s předponou `output` – výstupní soubory skriptu, které server úspěšně detekoval a pracoval s nimi při generování textu.

4.4 Autentizace uživatelů

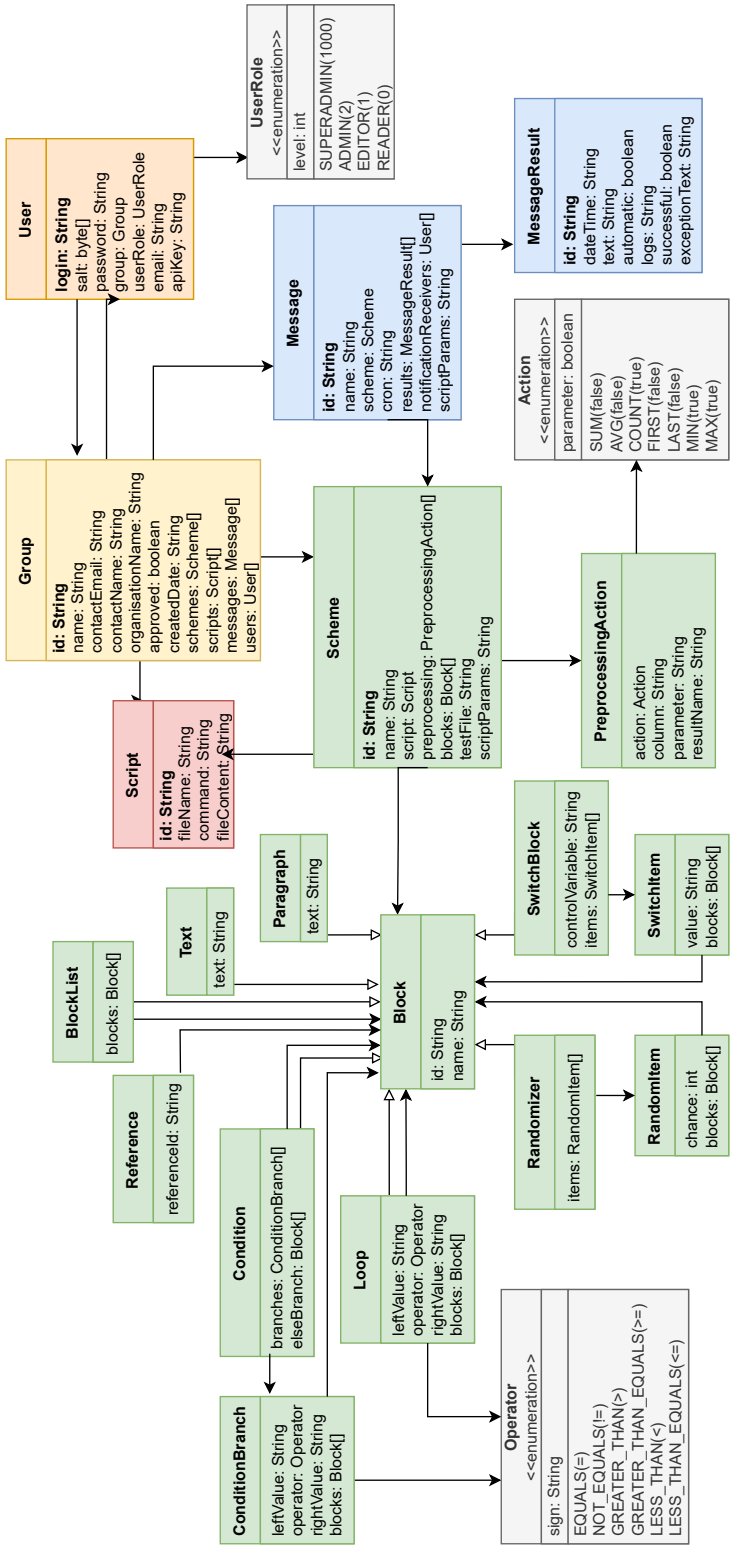
Každý uživatel je autentizován pomocí JWT tokenu. Tento token je vytvořen pomocí soukromého klíče serverové aplikace a obsahuje v jeho těle informace o přihlašovacím jméně daného uživatele, datu vydání tokenu a datu expirace tokenu.

4.4.1 Webová aplikace

Pro získání tokenu se musí uživatel přihlásit. Po úspěšné autentizaci uživatele pomocí jména a hesla obdrží webová aplikace od serveru token validní pro daného uživatele. Ve webové aplikaci je token uchováván v místním úložišti. V případě potřeby tokenu je z tohoto úložiště načten a přidán k požadavku odesílanému na server. V případě odhlášení uživatele je token z úložiště odstraněn. Při přihlášení uživatele je token opět do úložiště uložen.

4.4.2 Serverová aplikace

Serverová aplikace autentizuje uživatele pomocí jména a hesla pouze na rozhraní `authenticate`, které vyžaduje přihlašovací jméno a heslo. Po úspěšném přihlášení vytvoří přihlašovací token pro daného uživatele. Všechna ostatní rozhraní jsou zabezpečena tímto tokenem. Díky informacím uložených v tokenu je možné ověřit, o jakého uživatele se jedná. S touto informací následně server pracuje při autorizaci uživatele pro jednotlivé akce. Každý uživatel má navíc přidělen jeden token, který nemá danou dobu expirace a mění se pouze v případě změny hesla. Tento token může uživatel použít pro volání rozhraní například mimo webového klienta. Token si uživatel může zobrazit ve webové aplikaci na stránce s jeho profilem.



Obrázek 4.1: Databázový model

Instalace 5

Aplikace distribuována pomocí soukromého repositáře na platformě GitLab, ke kterému je možné udělit přístup v případě zájmu o instalaci aplikace. Zdrojové kódy aplikace odpovídající zmíněnému repositáři jsou také přílohou této práce. Repositář je potřeba příkazem `git clone <url> --recurse-submodules` naklonovat do prostředí, ve kterém bude aplikace spuštěna. Poté je potřeba v souboru `.env` doplnit údaje potřebné pro spuštění aplikace:

- `FE_BACKEND_PROTOCOL` – protokol (`http` nebo `https`),
- `FE_BACKEND_ADDRESS` – URL adresa nebo IP adresa serveru, na kterém je spuštěn server,
- `FE_BACKEND_PORT` – port, na kterém server naslouchá příchozím požadavkům (v případě přepsání výchozí hodnoty `8181`),
- `KEY_PRIVATE` – RSA soukromý klíč použitý k šifrování přístupových klíčů,
- `KEY_PUBLIC` – RSA veřejný klíč použitý k dešifrování přístupových klíčů,
- `MAIL_FROM` – (nepovinné) odchozí emailová adresa pro zasílání upozornění na vygenerované zprávy apod.,
- `MAIL_HOST` – (nepovinné) adresa SMTP serveru pro emailovou komunikaci,
- `MAIL_USER` – (nepovinné) přihlašovací jméno na SMTP server,
- `MAIL_PASSWORD` – (nepovinné) přihlašovací heslo na SMTP server,
- `MAIL_AUTH` – (nepovinné) zda je vyžadováno přihlašování k SMTP serveru (`true` nebo `false`),
- `SHARED_DATA_SPACE` – zda je povoleno využití sdíleného prostoru pro odkládání dat (`true` nebo `false`).

Po vyplnění těchto položek je potřeba pomocí příkazu `docker-compose build` aplikaci sestavit (sestaví se jak webový klient, tak server). Po sestavení aplikace je možné aplikaci příkazem `docker-compose up` spustit. Pokud spuštění neproběhne korektně, je nutné příkazem `docker volume create mongo_data` vytvořit obraz disku pro databázi a opakovat předchozí příkaz. Po prvním spuštění aplikace je žádoucí se přihlásit na uživatele Superadmin (přihlašovací jméno i heslo je `superadmin`), změnit tomuto uživateli heslo a nastavit emailovou adresu pro příjem upozornění na příchozí žádosti o založení nových skupin.

Aplikace je sestavena a spouštěna pomocí Dockeru. Pro webovou aplikaci a server je vytvořen samostatný obraz, který lze spustit buď samostatně, nebo jako celek pomocí konfiguračního souboru `docker-compose.yml`, který je umístěn v kořenovém adresáři projektu. Stejně je tomu tak i pro serverovou část.

5.1 Popis struktury Docker obrazů

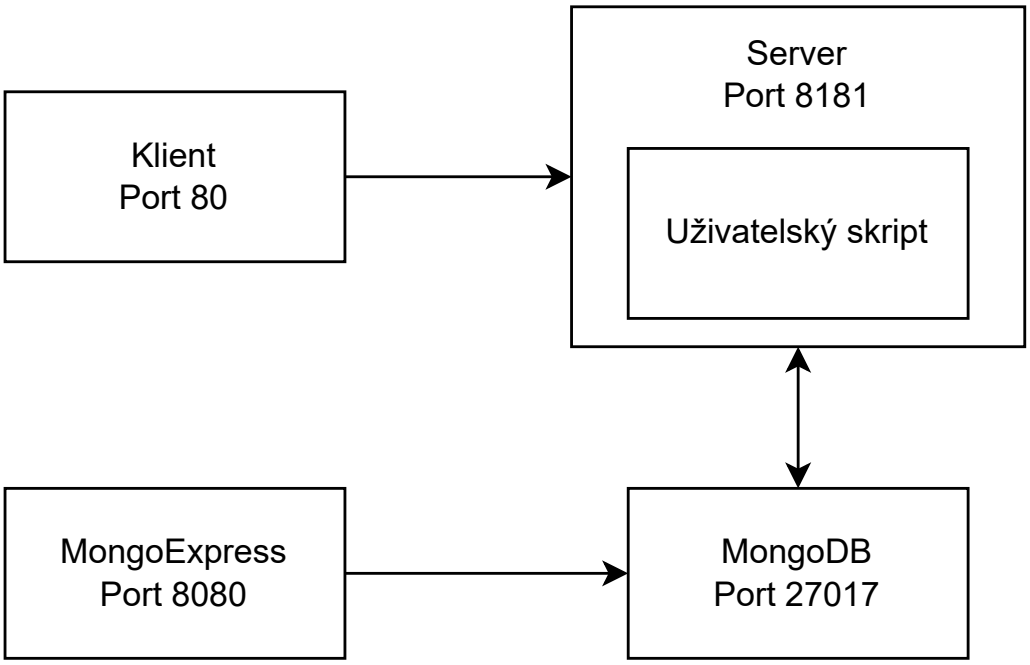
Pro úplné fungování aplikace je potřeba celkem tří obrazů (klient, server a databáze), ovšem pro snadný přístup k datům v databázi je využíván webový prohlížeč databáze MongoExpress, který je zabalen v samostatném obraze. Všechny tyto obrazy jsou propojeny do jedné sítě a mohou spolu tak komunikovat pouze pomocí jména obrazu a portu, na kterém je daný kontejner spuštěn. Architektura vytvořených Docker obrazů je naznačena na obrázku 5.1 na str. 41.

5.1.1 Webový klient

Obraz webového klienta je založen na obraze `nginx:alpine`, což je obraz webového serveru. Sestavení tohoto obrazu je konfigurováno v souboru `Dockerfile`. V prvním kroku se sestaví webová aplikace příkazem `npm ci && npm run-script build`. Ve druhém kroku se pak do obrazu `nginx:alpine` nakopírují přeložené zdrojové soubory z prvního kroku a spustí se webový server, který čeká na příchozí požadavky na portu `80`.

5.1.2 Server

Obraz serveru je založen na obraze `docker:dind`. Jedná se o obraz, který umožňuje spouštění Docker kontejnerů v rámci již spuštěného Docker kontejneru. Této funkcionality je zapotřebí pro spouštění uživatelsky definovaných skriptů v odděleném prostředí kvůli bezpečnosti. Sestavení obrazu serveru je konfigurováno v souboru `Dockerfile` a je opět rozděleno na dvě části. V první části jsou pomocí příkazu `mvn package` přeloženy zdrojové soubory aplikace. V druhém kroku je nejprve do obrazu `docker:dind` nainstalováno prostředí JDK, poté je do něj nakopírován spustitelný `.jar` soubor a soubory potřebné pro spouštění uživatelsky



Obrázek 5.1: Struktura Docker obrazů

definovaných skriptů. Poté je server, naslouchající příchozím požadavkům na portu `8181`, spuštěn.

5.1.2.1 Uživatelsky definované skripty

Pro spouštění uživatelsky definovaných skriptů je připraven soubor `Dockerfile` v kořeni projektu serveru v adresáři `docker-skript-data`. Tento soubor obsahuje návod pro sestavení obrazu pro exekuci Python skriptu, který je založen na obrazu `python:3.8-slim-buster`. Při sestavení obrazu jsou nejprve nainstalovány závislosti definované v souboru `requirements.txt`, poté je do obrazu nakopírován samotný skript, případně i testovací data. Samotný kontejner je poté spuštěn příkazem pro spuštění skriptu, který uživatel sám definuje.

5.1.3 Databáze

Pro databázi je použit obraz `mongo`. Tento obraz spustí databázi na portu `27017`. Pro zachování dat i při restartu kontejnerů je potřeba tomuto obrazu předat obraz disku, který se použije pro trvalé uchování dat. V tomto případě je to obraz disku s názvem `mongo_data`.

5.1.4 Prohlížeč databáze

Pro snadný přístup k datům v databázi a jejich prohlížení je používán obraz `mongo-express`. Ten na portu `8080` spustí webové rozhraní pro úpravu, tvorbu a čtení dat v databázi.

Testování systému 6

6.1 Přenesení existujícího systému

V rámci otestování aplikace byla přenesena funkcionalita systému, který vznikl v rámci projektu TAČR-TL02000288 a který který pomocí automatizace každý den generuje články o aktuálním dění na burze.

Pro úplnou replikaci funkcionality je nutné definovat skript pro generování stahování dat z burzy a jejich převedení do použitelné formy, šablonu pro výsledný text a zprávu pro automatické spouštění generování textu. Jelikož je počítáno s využíváním tohoto systému novináři, je detailní popis implementace uveden v samostatné příloze této práce. Níže budou pouze zběžně popsány jednotlivé implementace.

6.1.0.1 Skript

Jako skript byl použit upravený skript, který v rámci projektu TAČR-TL02000288 generoval novinové články. Tento skript nyní stáhne data z burzy, transformuje je do potřebné podoby a uloží do výstupních souborů:

- `output_ak.csv` – základní informace o akciích,
- `output_ak_sorted.csv` – oindexované základní informace o akciích seřazené podle změny hodnoty akcie vzestupně,
- `output_ak_sorted_rev.csv` – oindexované základní informace o akciích seřazené podle změny hodnoty akcie sestupně,
- `output_ak_obj_sorted.csv` – oindexované základní informace o akciích seřazené podle objemu obchodů sestupně,
- `output_bb.csv` – základní informace o akciích pro všechny indexy Pražské burzy,
- `output_bb_full.csv` – rozšířené informace o akciích pro všechny indexy Pražské burzy,

- `output_bi.csv` – informace o všech indexech Pražské burzy.

Skript může přijmout jeden parametr, ten značí počet dnů, o který se při generování posuneme do minulosti. Pokud tedy skriptu předáme jako parametr číslo 1, budeme pro generování používat včerejší data.

6.1.0.2 Šablona

Šablona byla implementována tak, aby odpovídala diagramům, podle kterých byl generován článek v projektu TAČR-TL02000288. Detailní popis je uveden v příloze.

6.1.0.3 Zpráva

Zpráva byla nadefinována tak, aby v předem určený čas automaticky spustila generování textu. Cron výraz je nastaven na hodnotu `1 30 17 * * MON-FRI`, tím je generování textu spuštěno každý pracovní den v 17 hodin a 30 minut. Výsledky generování se automaticky ukládají pod námi definovanou zprávu.

6.2 Jednotkové testování

Pro otestování zdrojového kódu serverové aplikace byla zvolena metoda jednotkového testování. Pro tento účel byla využita knihovna `JUnit` verze `5.9.2`. Aby bylo možné otestovat interakci s databází bez nutnosti použití skutečné databáze, byla využita vestavěná databáze MongoDB za použití knihovny `de.flapdoodle.embed.mongo` verze `4.6.2`.

6.2.1 Pokrytí kódu jednotkovými testy

Celkem bylo jednotkovými testy pokryto 96 % tříd, 98 % metod a 95 % řádků zdrojového kódu aplikace. Jednotkové testy byly psány pro všechny třídy aplikace kromě tříd, které dědí od třídy `java.lang.Exception`, ty byly však z 84 % procent pokryty ostatními jednotkovými testy. Celkový přehled pokrytí kódu pro jednotlivé balíky je zaznamenán v tabulce 6.1 na str. 45.

6.3 Studie efektivity

Pro otestování rychlosti generování textu bylo připraveno šest scénářů testujících odlišná nastavení, šablony a skripty. Pro každý scénář bylo provedeno deset měření. Z výstupů všech měření pak byly vypočítány průměrné, nejnižší a nejvyšší časy.

Balík	Procento pokrytí tříd	Procento pokrytí metod	Procento pokrytí řádek kódu
com.julrych.data2text	96 %	98 %	95 %
c.j.d.api	100 %	100 %	99 %
c.j.d.config	100 %	100 %	66 %
c.j.d.data	100 %	100 %	100 %
c.j.d.enums	100 %	100 %	100 %
c.j.d.exceptions	84 %	84 %	84 %
c.j.d.functions	100 %	100 %	100 %
c.j.d.model	100 %	100 %	100 %
c.j.d.repositories	100 %	100 %	100 %
c.j.d.response	100 %	100 %	100 %
c.j.d.service	100 %	96 %	91 %
c.j.d.utils	100 %	100 %	100 %

Tabulka 6.1: Přehled pokrytí kódu jednotkovými testy pro jednotlivé balíky

6.3.1 Měřené scénáře

6.3.1.1 Burza bez sdíleného prostoru dat

Tento scénář generuje zprávu popisující dění na burze v daný den. Skript použitý pro generování dat musí stáhnout všechna data z burzy, poté je zpracovat a uložit do výsledných souborů. Šablona pro generování textu je poměrně složitá. Navíc aplikace nedovoluje ukládání dat do sdíleného prostoru. Skript tedy pro každé generování textu musí data stahovat znovu. Cílem je otestovat kombinaci komplexního skriptu a šablony.

6.3.1.2 Burza se sdíleným prostorem dat

Skript i šablona jsou stejné jako v případě výše. Aplikace ale dovoluje ukládání dat do sdíleného prostoru. Skript má tedy data stažená ze serveru burzy dostupná z předchozích generování. Cílem je otestovat kombinaci komplexního skriptu a šablony a porovnat časy s generováním textu bez použití sdíleného prostoru pro data.

6.3.1.3 Data z FTP serveru

V tomto scénáři skript stahuje jeden `csv` soubor o 100 řádcích z FTP serveru, přidá k němu dva vypočtené sloupce a uloží jej do výstupního souboru. Šablona pak jedním cyklem projde všechna data a pro každou řádku vygeneruje krátký text. Cílem je otestovat generování textu pro malá data získaná z externích zdrojů.

6.3.1.4 Data z testovacího souboru

Výstupní soubor skriptu i šablona je stejná jako v předchozím případě, vstupní soubor skriptu je ale převzat od uživatele pomocí předaného testovacího souboru (ten je stejný jako soubor nahraný na FTP serveru z předchozího scénáře). Skript se tedy nemusí na FTP server připojovat. Cílem je otestovat generování textu pro malá data získaná z lokálních zdrojů.

6.3.1.5 Velká výstupní data

Skript i šablona jsou použity stejné jako v předchozích dvou scénářích. Vstupní soubor je předán uživatelem pomocí testovacího souboru. Rozdílem je velikost vstupního (a tedy i výstupního) souboru, ten nyní obsahuje 10000 záznamů. Cílem je otestovat generování textu pro velká data.

6.3.1.6 Žádná vstupní data

V tomto scénáři je použit skript, který nevyžaduje žádná vstupní data. Generuje výstupní soubor o 10 řádcích, které jsou nadefinovány přímo v těle skriptu. Šablona poté generuje pouze statický text. Cílem je otestovat generování textu pro absolutně minimalistický případ.

6.3.2 Měřené časy

Pro každý scénář byly měřeny 4 časy.

6.3.2.1 Příprava Docker prostředí

Tento čas odpovídá celkové době, po kterou je připravován Docker obraz pro běh skriptu. Zahrnuje tedy načtení základního obrazu, kopírování souborů do obrazu, instalaci Python závislostí, sestavení obrazu a jeho spuštění. Tento čas je při opakovaném spouštění výrazně zkrácen, jelikož Docker dokáže jednotlivé kroky sestavení obrazu ukládat do mezipaměti a znovu je používat. Tento čas je podmnožinou celkového času exekuce skriptu.

6.3.2.2 Celkový čas exekuce skriptu

Odpovídá celkové době, po kterou byla skriptem generována výstupní data. Čas tedy zahrnuje přípravu Docker prostředí a je hlavně ovlivňován obsahem skriptu.

6.3.2.3 Čas vyhodnocení šablony

Tento čas odpovídá době, po kterou byla vyhodnocována šablona a generován text. Čas může být ovlivněn strukturou šablony a také velikostí dat generovaných skriptem.

6.3.2.4 Celkový čas (skript + generování)

Tento čas odpovídá celkové době generování textu. Je v něm tedy zahrnuta jak doba běhu skriptu, tak doba vyhodnocení šablony.

6.3.3 Výsledky měření

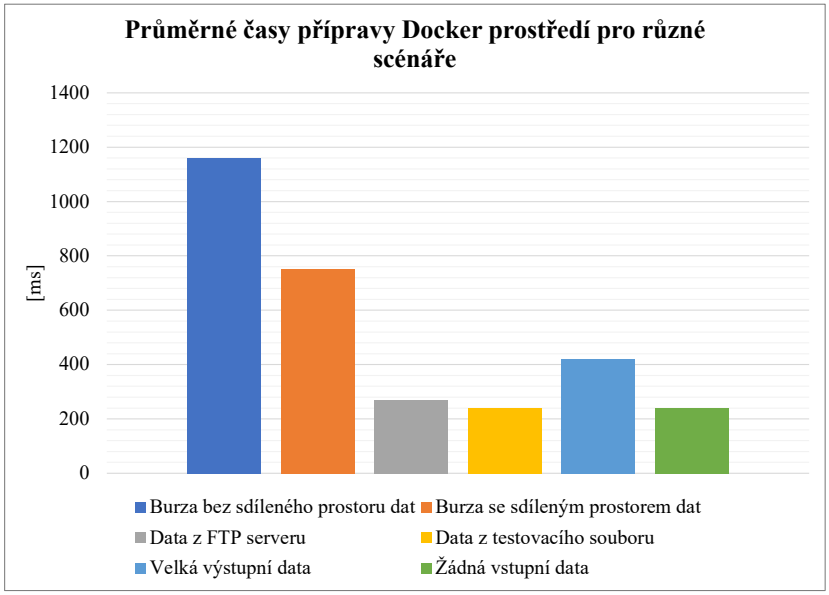
Celkem bylo provedeno 60 měření časů běhu programu. Tato měření byla rozdělena do kategorií podle testovacího scénáře a jednotlivých částí generování. Vypočítané průměrné časy jsou uvedeny v tabulce 6.2 na str. 49, nejnižší naměřené časy pak v tabulce 6.3 na str. 49 a nejvyšší naměřené časy v tabulce 6.4 na str. 51.

6.3.3.1 Příprava Docker prostředí

Příprava prostředí pro vykonání skriptu zabere v průměru 240 až 1160 milisekund. V ideálním případě, kde jsou všechny kroky tvorby Docker obrazu uloženy v mezipaměti, nezabere tento krok žádný čas. V nejhorším případě, kdy v mezipaměti žádný krok uložen není, může příprava zabrat až 3600 milisekund. Z grafu 6.1 na str. 48 je vidět, že pro scénáře s vyšší výpočetní náročností (burza, velký vstupní soubor) jsou časy přípravy vyšší.

6.3.3.2 Celkový čas exekuce skriptu

Doba exekuce skriptu je absolutně závislá na daném skriptu. Průměrné časy exekuce skriptu pro jednotlivé scénáře jsou znázorněny v grafu 6.2 na str. 50. Nejnižší čas exekuce skriptu byl naměřen s hodnotou 1750 milisekund. Nižší hodnoty bude již velmi těžké dosáhnout, jelikož z času přípravy prostředí je vidět, že všechny kroky tvorby Docker obrazu byly uloženy v mezipaměti. Zbývající čas tedy odpovídá startu Docker kontejneru, získání vygenerovaných dat a ukončení Docker kontejneru se skriptem. Z porovnání scénářů burzy se sdíleným prostorem pro uložení dat a bez něj je vidět, že při použití sdíleného prostoru se průměrný čas snížil o více než dvě třetiny. Použití sdíleného prostoru pro data tedy může ve specifických případech výrazně urychlit dobu běhu skriptu.



Obrázek 6.1: Grafické znázornění průměrného času přípravy Docker prostředí pro různé scénáře

6.3.3.3 Čas vyhodnocení šablony

Čas potřebný pro vyhodnocení skriptu je závislý na konkrétní šabloně a vygenerovaných datech. Průměrné časy vyhodnocení šablony jsou znázorněny v grafu 6.3 na str. 50. Z grafu je vidět, že tento čas závisí hlavně na vstupních datech, kdy pro scénář s daty o 10000 řádcích byl tento čas mnohonásobně vyšší, než pro ostatní scénáře.

6.3.3.4 Celkový čas (skript + generování)

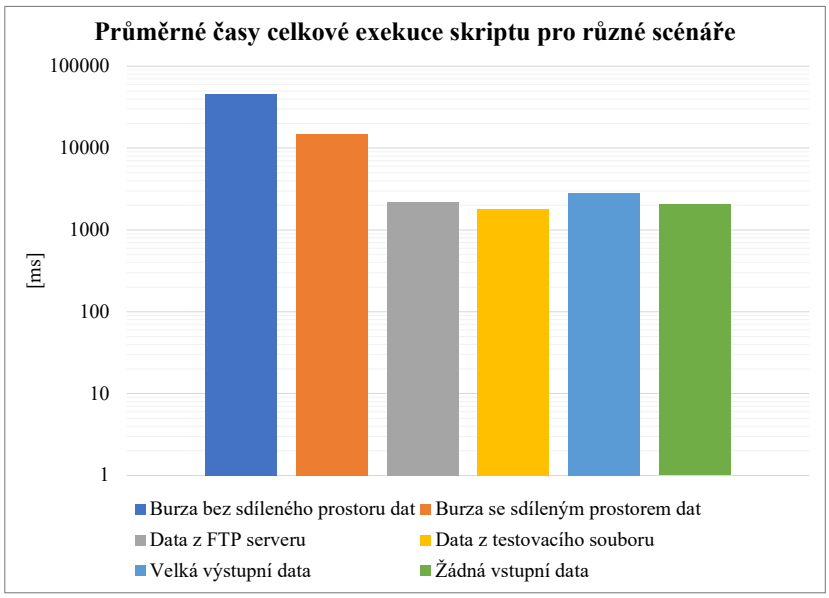
Z grafu 6.4 na str. 51 je vidět, že průměrný čas generování textu pro jednotlivé scénáře přibližně kopíruje průměrné časy běhu skriptu. Je tedy patrné, že nejvyšší vliv na celkovou dobu generování textu má právě skript a jeho efektivita.

Scénář	Příprava Docker prostředí	Celkový čas exekuce skriptu	Čas vyhodnocení šablony	Celkový čas (skript + generování)
Burza bez sdíleného prostoru dat	1160 ms	45417 ms	18 ms	45436 ms
Burza se sdíleným prostorem dat	750 ms	13749 ms	11 ms	14760 ms
Data z FTP	270 ms	2179 ms	3 ms	2183 ms
Data z testovacího souboru	240 ms	1819 ms	3 ms	1822 ms
Velká výstupní data	420 ms	2843 ms	243 ms	3087 ms
Žádná vstupní data	240 ms	2051 ms	1 ms	2052 ms

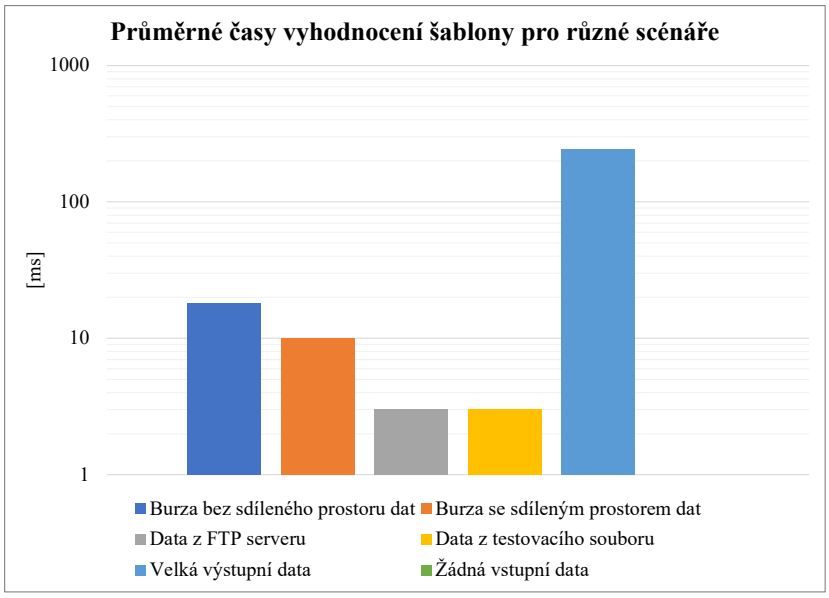
Tabulka 6.2: Průměrné časy generování textu pro různé scénáře

Scénář	Příprava Docker prostředí	Celkový čas exekuce skriptu	Čas vyhodnocení šablony	Celkový čas (skript + generování)
Burza bez sdíleného prostoru dat	0 ms	42325 ms	7 ms	42343 ms
Burza se sdíleným prostorem dat	0 ms	13331 ms	5 ms	13340 ms
Data z FTP	0 ms	1926 ms	2 ms	1929 ms
Data z testovacího souboru	0 ms	1693 ms	2 ms	1695 ms
Velká výstupní data	0 ms	2681 ms	211 ms	2892 ms
Žádná vstupní data	0 ms	1750 ms	1 ms	1751 ms

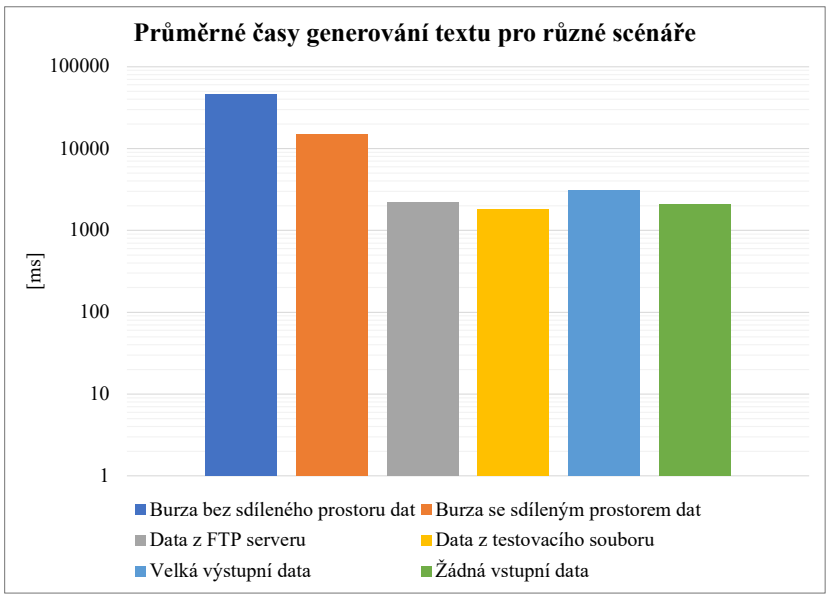
Tabulka 6.3: Nejnižší časy generování textu pro různé scénáře



Obrázek 6.2: Grafické znázornění průměrného času vykonání skriptu pro různé scénáře



Obrázek 6.3: Grafické znázornění průměrného času vyhodnocení šablony pro různé scénáře



Obrázek 6.4: Grafické znázornění průměrného času celkového generování textu pro různé scénáře

Scénář	Příprava Docker prostředí	Celkový čas exekuce skriptu	Čas vyhodnocení šablony	Celkový čas (skript + generování)
Burza bez sdíleného prostoru dat	3600 ms	52628 ms	49 ms	52643 ms
Burza se sdíleným prostorem dat	3500 ms	19578 ms	18 ms	19595 ms
Data z FTP	900 ms	3328 ms	5 ms	3333 ms
Data z testovacího souboru	500 ms	2201 ms	4 ms	2205 ms
Velká výstupní data	500 ms	3062 ms	301 ms	3283 ms
Žádná vstupní data	500 ms	2858 ms	1 ms	2859 ms

Tabulka 6.4: Nejvyšší časy generování textu pro různé scénáře

Závěr 7

Cílem této práce bylo seznámit se s problematikou automatického generování zpráv ze strukturovaných dat, navrhnout systém, který bude dle uživatelsky definovatelných šablon v daných časech automaticky generovat zprávy, takový systém implementovat a otestovat jej přenesením funkcionality z existujícího systému.

V první části práce byly shrnuty poznatky novinářů z práce s automatickým generováním textu a popsány výhody a nevýhody tohoto přístupu k psaní článku. Z těchto poznatků pak byly definovány požadavky na nový systém, jako například různorodost zdrojů dat, transformace získaných dat a intuitivní nástroj pro definování šablon, podle kterých bude výsledný text generován.

Podle zjištěných požadavků byl pak navržen systém, který by měl tyto požadavky splňovat. Jako řešení různorodosti dat bylo navrženo použití skriptu, který uživateli umožní získání dat z jakéhokoli zdroje. Použitím skriptu byl také vyřešen požadavek na transformaci dat, jelikož tu je v rámci skriptu také možné provést. Pro jednoduchou analýzu dat pak byl navržen systém akcí pro předzpracování dat, jenž je ke skriptu doplňkem. Nástroj pro tvorbu šablon byl pak navržen tak, aby reflektoval myšlení novinářů při psaní článku. V nástroji se používají jednoduché bloky, které svými vlastnostmi napodobují základní struktury z programovacích jazyků. Bloky jsou pak v šabloně, stejně jako novinový článek, stavěny odshora dolů.

V praktické části pak byla popsána implementace navrženého systému. Byla popsána konkrétní řešení jednotlivých problémů, jako například výběr skriptovacího jazyka pro uživatelsky definovatelné skripty, nebo implementovaná rozhraní. Také byla popsána instalace systému.

Systém byl testován přenesením funkcionality z existujícího systému do systému nového. Generování zpráv o dění na burze bylo do nového systému úspěšně přeneseno. Jako příloha této práce vznikl návod, který mohou novináři použít jako inspiraci pro tvorbu vlastních automaticky generovaných zpráv. Zdrojové kódy aplikace byly otestovány jednotkovými testy. V poslední části práce byla také studována efektivita systému. Ten vykázal velmi dobré výsledky, kdy v nejlepším případě byl schopen generovat zprávu o dění na burze do čtvrt minuty, zrychlení v porovnání s ručním psaní textu je tak mnohonásobné. Texty, jež byly založeny na jednoduš-

ších šablonách, pak byly generovány v řádech jednotek sekund. Systém tedy splňuje požadavek na intuitivní tvorbu šablon, také se dokáže dvěma nástroji vypořádat s různorodými zdroji dat a jejich transformací, systém také umožňuje automatické spouštění generování textu a ukládání vygenerovaných textů.

Systém je možné využít pro různou škálu reportů. Funkce systému byla demonstrována na burzovním zpravodajství, je jej ale možné využít například i pro generování textů o počasí, rychlých zpráv ze sportovních utkání a v dalších oblastech, ve kterých je psaní zprávy založené hlavně na interpretaci dat.

Přehled zkratek

API	Application Programming Interface
CSS	Cascading Style Sheets
CSV	Comma Separated Values
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifikace
IP	Internet Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
MVC	Model View Controller
OOP	objektově orientované programování
PSE	Prague Stock Exchange
REST	Representational State Transfer
RSA	Rivest, Shamir, Adleman
SMPT	Simple Mail Transfer Protocol
TAČR	Technologická agentura České republiky
URL	Uniform Resource Locator

Bibliografie

1. MONTAL, Tal; REICH, Zvi. I, Robot. You, Journalist. Who is the Author? *Digital Journalism*. 2017, roč. 5, č. 7, s. 829–849. Dostupné z DOI: 10.1080/21670811.2016.1209083.
2. MORAVEC, Václav; MACKOVÁ, Veronika; SIDO, Jakub; EKŠTEIN, Kamil. The Robotic Reporter in the Czech News Agency: Automated Journalism and Augmentation in the Newsroom. *Communication Today*. 2020, roč. 11, č. 1, s. 36. Dostupné také z: https://communicationtoday.sk/wp-content/uploads/03_MORAVEC-et-al_CT-1-2020.pdf.
3. ČTK. *ČTK při volbách poprvé využila automatické zprávy*. 2018. Dostupné také z: https://www.ctk.cz/novinky/?id=2764#.
4. MICROSOFT. *Co je relační databáze?* 2023. Dostupné také z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-a-relational-database/#whatis. Slovník cloudových výpočetních operací.
5. ZÍMA, Martin. *NoSQL databáze*. 2020. Dostupné také z: https://www.kiv.zcu.cz/studies/predmety/db2/cviceni-nosql.html.
6. HEMEL, Zef. *Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews*. Media Inc., 2013. Dostupné také z: https://www.infoq.com/news/2013/06/facebook-react/.
7. HÁMORI, Ferenc. *The History of React.js on a Timeline*. RisingStack Inc, 2022. Dostupné také z: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/.
8. GOOGLE. *Introduction to Angular concepts*. 2022. Dostupné také z: https://angular.io/guide/architecture.
9. GAVIGAN, Dave. *The History of Angular*. 2018. Dostupné také z: https://medium.com/the-startup-lab-blog/the-history-of-angular-%5C-3e36f7e%5C-828c7.

10. SHARMA, Anubhav. *What Is Express JS In Node JS?* 2023. Dostupné také z: `https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js`.
11. MICROSOFT. *Co je Java Spring Boot?* 2023. Dostupné také z: `https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-java-spring-boot/`. Slovník cloudových výpočetních operací.

Seznam obrázků

Seznam tabulek

Seznam výpisů

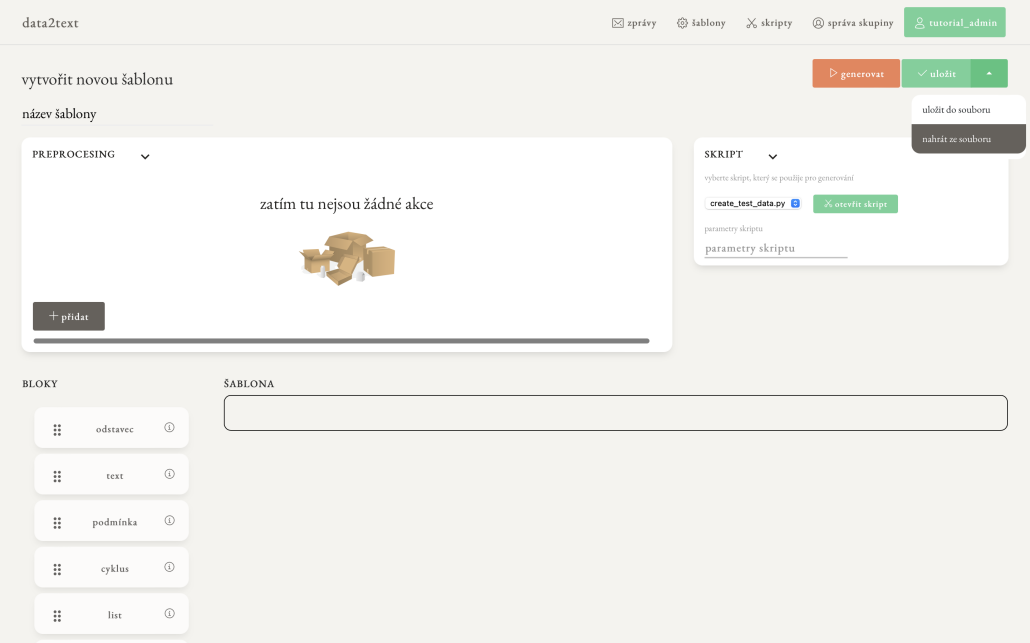
Uživatelská příručka

A.1 Šablony

Pomocí šablon se data vygenerovaná skriptem převedou do textové podoby.

A.1.0.1 Ovládací prvky

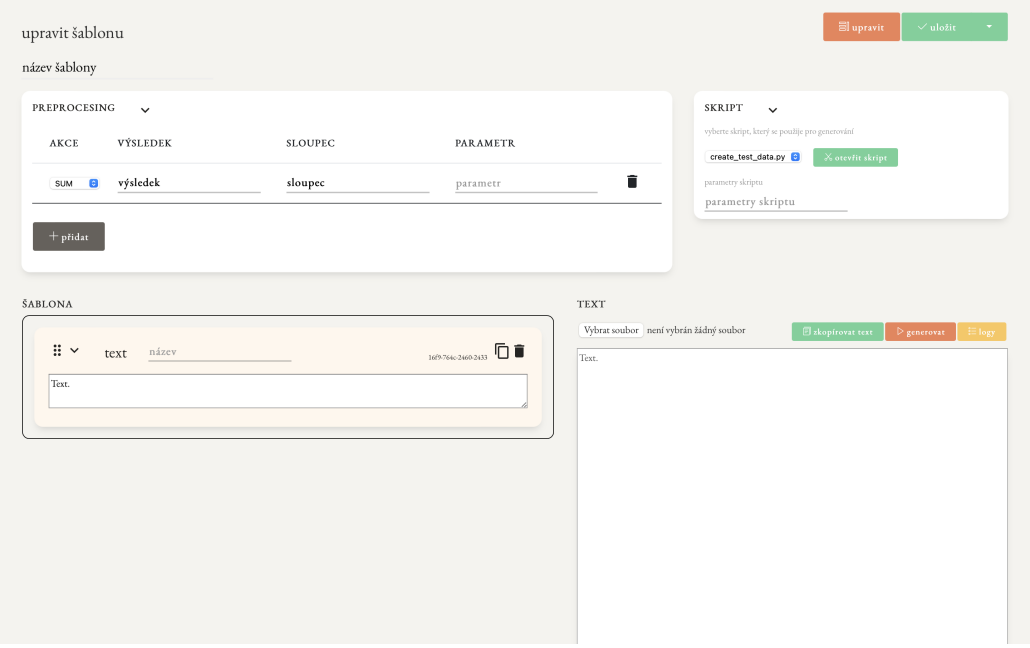
- Pod nadpisem je možné vyplnit název šablony.
- Vpravo nahoře se nachází tlačítko pro generování textu a uložení šablony, po rozkliknutí šipky u tlačítka uložit je možné uložit šablonu do souboru nebo ji ze souboru nahrát.
- Pod názvem šablony se nachází tabulka pro definování akcí pro předzpracování dat. Kliknutím na tlačítko „přidat“ můžeme přidat novou akci pro předzpracování dat.
- Vedle tabulky s akcemi pro předzpracování dat se nachází tabulka pro zvolení skriptu. Zde si můžeme vybrat skript, který se pro generování použije, otevřít ho a předat mu parametry.
- Ve spodní části vlevo se pak nachází seznam všech použitelných bloků, které můžeme přetažením umístit do šablony.
- V poli pro šablonu už pak definujeme samotnou strukturu šablony.

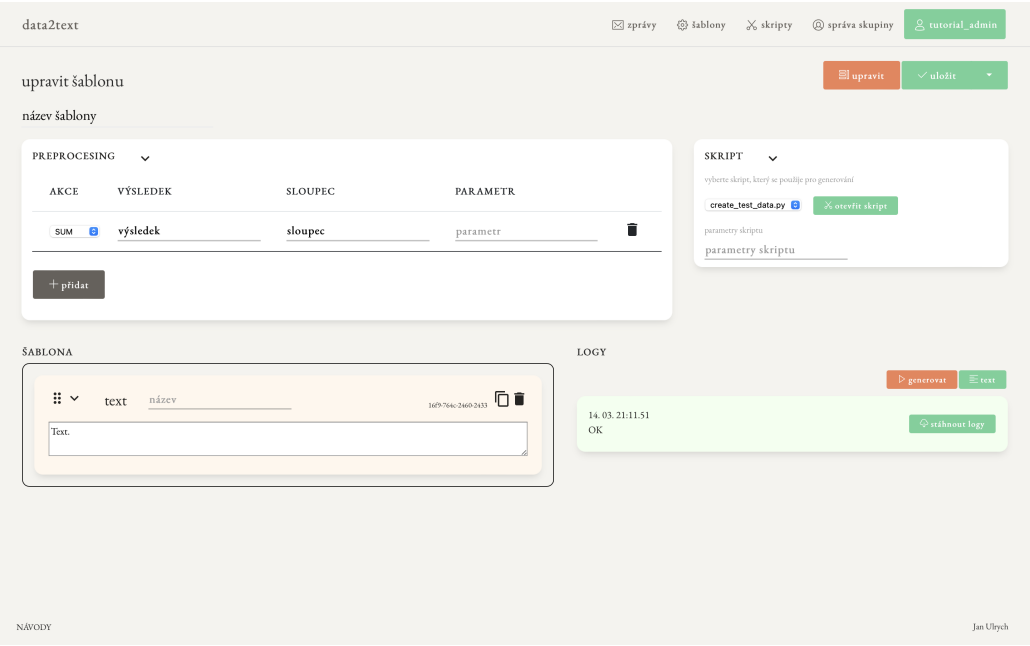


A.1.0.2 Generování textu

Po kliknutí na tlačítko generovat se změní rozložení obrazovky. Na levé straně je pole pro šablonu a na pravé pole s vygenerovaným textem.

- Můžeme vložit testovací soubor, který poté skript zpracuje.
- Kliknutím na tlačítko „zkopírovat text“ se do schránky vloží vygenerovaný text.
- Tlačítkem „generovat“ se spustí generování textu.
- Tlačítkem „logy“ se přepneme do zobrazení logů. Zde si můžeme prohlédnout výsledky posledního generování zpráv a případně stáhnout zip soubor s informacemi o daném generování.





A.1.0.3 Bloky

Bloky jsou základními stavebními prvky šablony. Každý z bloků má svou jedinečnou funkci a slouží k něčemu jinému. Blok můžeme do šablony přidat přetažením ze seznamu bloků do oblasti šablony. Každý blok má několik ovládacích prvků a svou jednoznačnou barvu:

- Ikona šesti teček slouží jako uchycení pro tažení bloku. Při držení tlačítka myši na této ikoně můžeme blokem pohybovat a přemístit ho.
- Ikona šipky směřující dolů slouží pro zabalení bloku. Po kliknutí na tuto ikonu obsah bloku zmizí. To může být užitečné, když máme větší šablonu a některé části šablony chceme na chvíli schovat.
- Vedle šipky se pak nachází obecný název bloku.
- Vedle obecného názvu bloku pak můžeme definovat ještě vlastní název bloku. Na ten se pak můžeme spolu s ID bloku odkazovat.
- Úplně vpravo se pak nachází ikona odpadkového koše. Kliknutím na tuto ikonu můžeme blok a celý jeho obsah smazat.
- Vedle ikony odpadkového koše se nachází ikona pro zkopírování ID bloku. Na ID bloku se můžeme odkazovat, pokud se nechceme odkazovat na název bloku.
- Nakonec vedle tlačítka pro zkopírování ID bloku najdeme ještě samotné ID bloku. To může být užitečné, pokud chceme bloky podle ID vyhledávat.

Následují sekce popisující jednotlivé bloky, pro všechny návody byl použit skript generující data zobrazená na obrázku A.1.0.3 na str. 68.

Obrázek A.1: Data použitá v návodech

id	jmeno	prijmeni	vek	profese	bmi	zmena_bmi	pohlavi
0	Eve	Blisse	32	policista	35.12	3.16	F
1	Melodie	Korey	60	policista	15.1	0.41	F
2	Belinda	Hirsch	72	policista	20.05	0.9	F
3	Suzette	Bethany	17	doktor	34.46	-0.85	F
4	Emmey	Seessel	44	doktor	38.95	1.22	F
5	Brietta	Orelee	48	hasič	40.42	-4.42	F
6	Ofilia	Longfellow	32	doktor	26.21	3.49	F
7	Carolina	Yerkovich	45	doktor	15.14	-3.56	F
8	Coral	Pelagias	29	policista	19.99	0.69	M
9	Dale	Suanne	52	doktor	19.34	0.1	M

A.1.1 Textové bloky

Textové bloky obsahují jedno textové pole, do kterého je možné zadat text, jenž bude tvořit nějakou část výsledné zprávy. K odstavci je na rozdíl od textu automaticky přidán znak konce řádky. Navazující text tedy bude začínat na nové řádce.

A.1.1.1 Vkládání hodnot proměnných

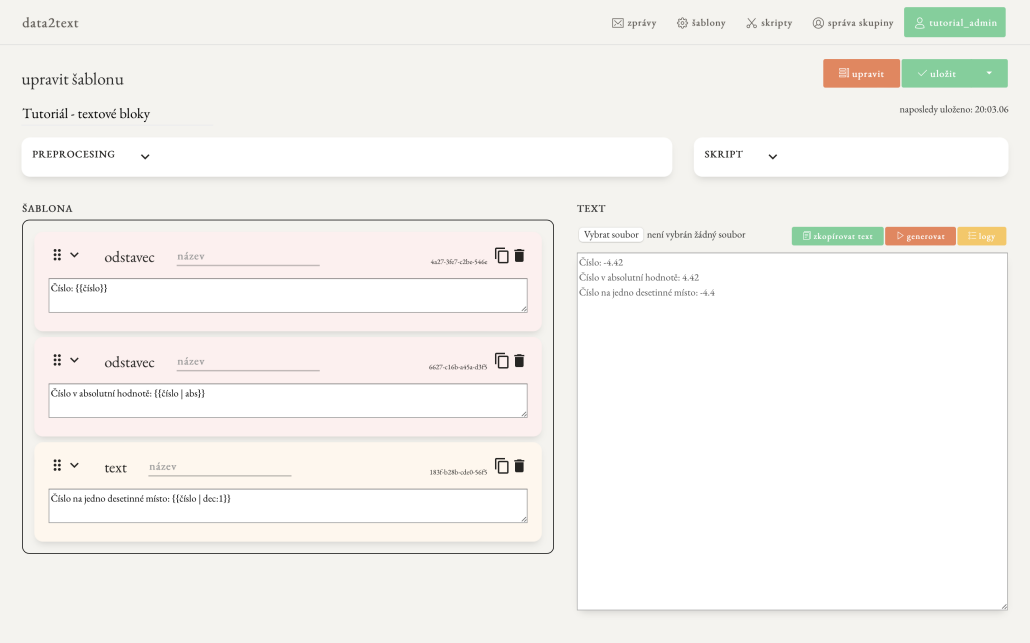
Hodnoty proměnných se dají do textu vložit pomocí složených závorek, tedy např. `{{proměnná_X}}`. Tato proměnná může být buď výsledkem akce pro předzpracování dat nebo hodnota proměnné v aktuální iteraci cyklu.

A.1.1.2 Formátování číselných proměnných

Proměnné, jenž mají číselnou hodnotu, je možné formátovat pomocí dvou rour: abs a dec. Roura abs do textu vloží absolutní hodnotu z číselné proměnné. Roura dec pak do textu vloží číslo s daným počtem desetinných míst. Pro každou proměnnou je možné použít pouze jednu rouru.

A.1.1.3 Příklad

Vyzkoušíme si použití vkládání hodnot proměnných, roury abs a roury dec.

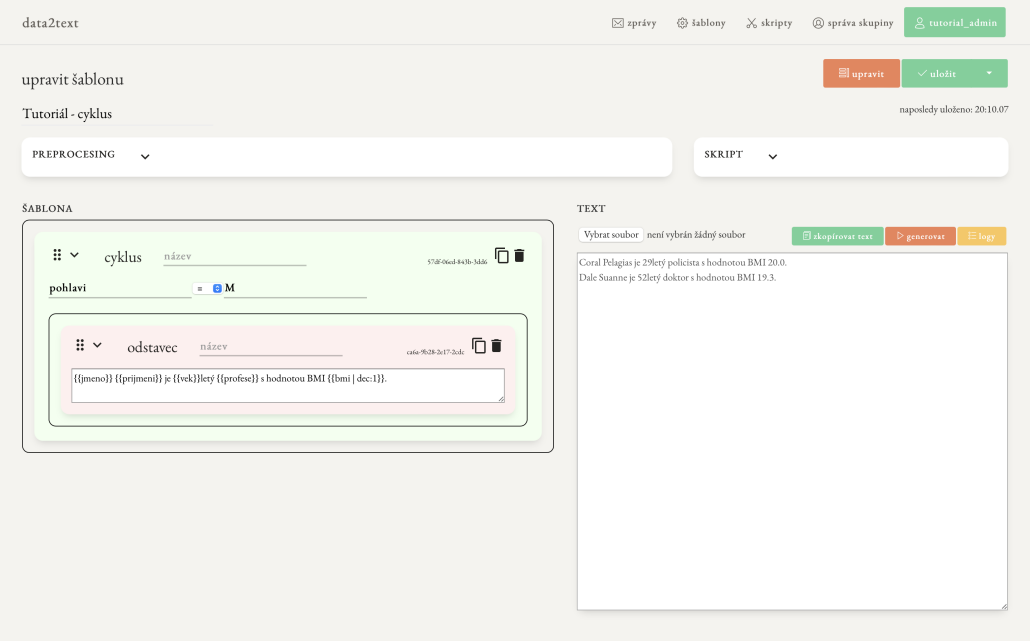


A.1.2 Cyklus

Cyklus postupně všechny řádky a porovnává obsah proměnné na levé straně výrazu s pravou stranou výrazu. Na levé straně musí být vždy nějaká proměnná (reprezentována názvem sloupce v CSV datech). Na pravé straně může být výsledek akce pro předzpracování dat, konstanta nebo další proměnná. Pokud je toto porovnání úspěšné, vykonají se bloky definované v těle cyklu.

A.1.2.1 Příklad

Chceme vypsat informace o všech mužích. Použijeme tedy cyklus s podmínkou „pohlavi = M“. Cyklus projde všechny řádky v datech a tam, kde je podmínka splněna, vyhodnotí bloky v jeho těle. Jelikož jsou v datech dva muži, výsledný text obsahuje dvě řádky.

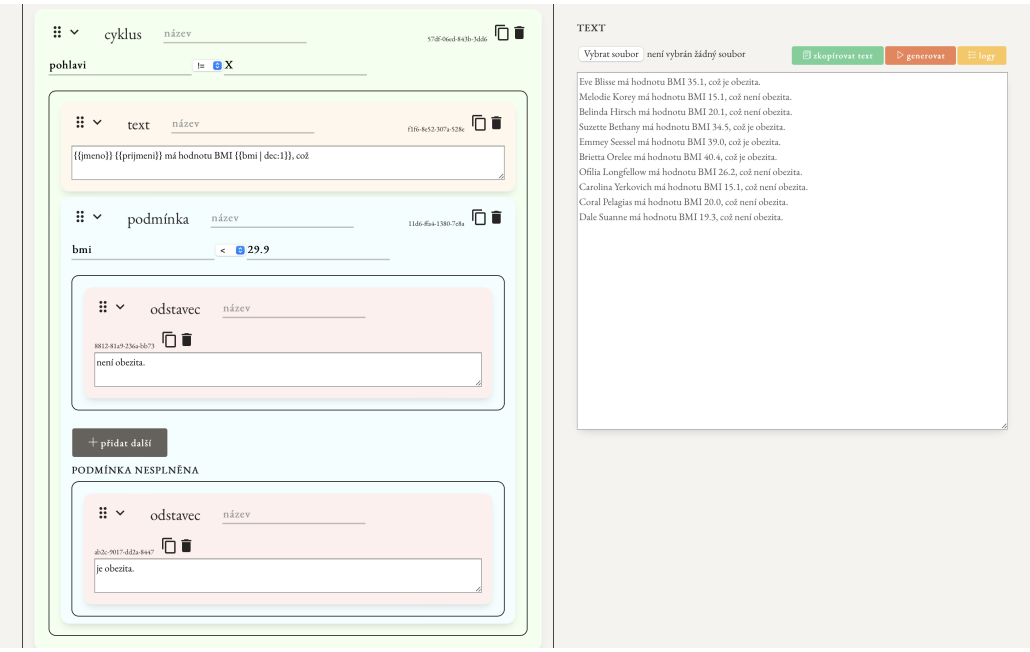


A.1.3 Podmínka

Podmínka může mít více větví, pro každou větev se vyhodnocuje levá strana s pravou stranou na základě definovaného operátoru. Na levé i právě straně mohou být jakékoli výrazy. Pokud je vyhodnocení pro nějakou větev úspěšné, vyhodnotí se bloky v dané větvi a podmínka končí, další větve se již nevyhodnocují. Pokud není vyhodnocena žádná větev, automaticky se vyhodnocuje větev „Podmínka nesplněna“. Další větve je možné přidávat tlačítkem „přidat další“.

A.1.3.1 Příklad

Chceme vypsat informaci o tom, jestli hodnota BMI u daného člověka znamená obezitu, či nikoli. Tuto informaci chceme vypsat pro všechny lidi, a tak použijeme podmínku „pohlavi != X“, ta totiž bude splněna pro všechny řádky v datech a bloky se tak vyhodnotí vždy. V samotném podmínkovém bloku použijeme podmínku „bmi < 29.9“ a dle výsledku této podmínky vypíšeme informace o obezitě.

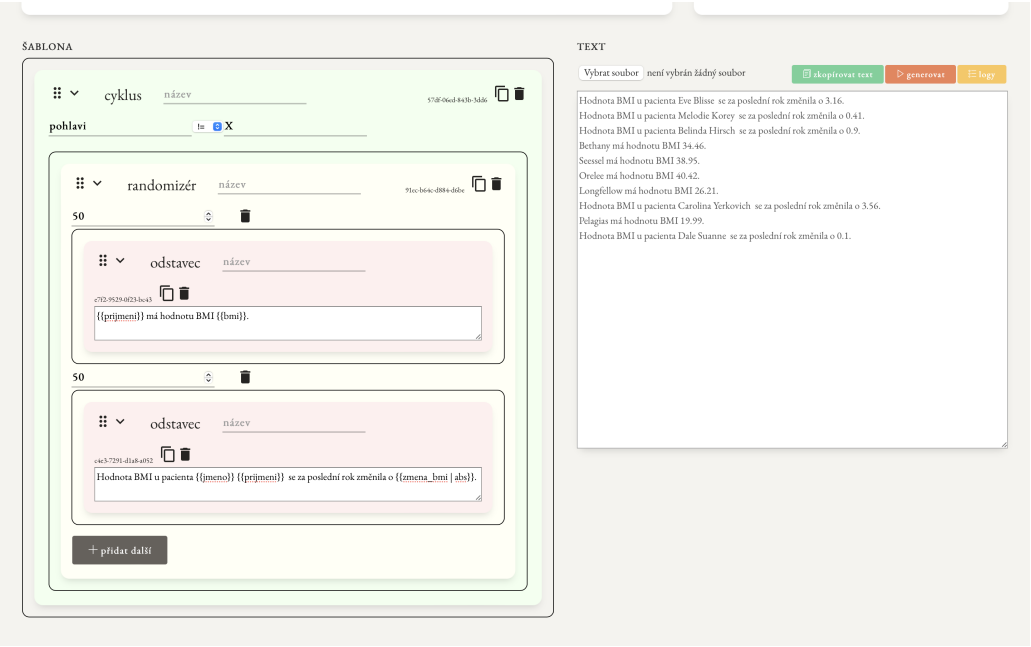


A.1.4 Randomizér

Randomizér náhodně vyhodnotí bloky se zadanou šancí. Šance pro všechny větve nemusí v součtu dávat žádné konkrétní číslo. Pokud se jedna větev vyhodnotí, další větve již vyhodnoceny nejsou. Další větve je možné přidávat tlačítkem „přidat další“.

A.1.4.1 Příklad

Pro každou osobu chceme náhodně vypsat jiné informace. Použijeme proto cyklus s podmínkou „pohlavi != X“, ta bude splněna vždy a informace vypíšeme o všech. Obsah informace ale pomocí randomizéru randomizujeme. S 50% šancí vypíšeme pouze hodnotu BMI a s 50% šancí vypíšeme změnu hodnoty BMI.

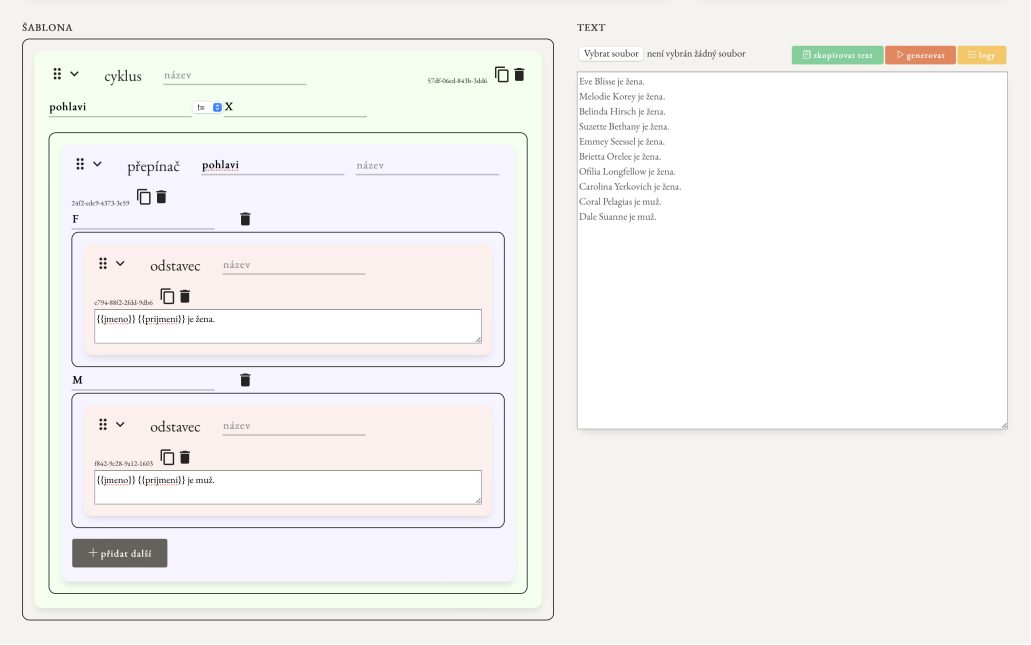


A.1.5 Přepínač

Přepínač funguje podobně jako podmínka, s tím rozdílem, že se vždy vyhodnocuje hodnota řídící proměnné s hodnotou konkrétní větve operátorem „=“. Pokud se jedna větev vyhodnotí, další již vyhodnocovány nejsou. Další větve je možné přidávat tlačítkem „přidat další“.

A.1.5.1 Příklad

Chceme vypsat pouze jednoduchou informaci o tom, zda je daná osoba muž nebo žena. K tomu použijeme přepínač v cyklu s podmínkou „pohlavi != X“ (ta bude splněna vždy). Řídící proměnná přepínače bude „pohlavi“. Jednotlivé větve přepínače pak budou mít hodnotu „F“ nebo „M“ a podle nich vypíšeme příslušný text. Pokud bychom chtěli tento samý výraz přepsat do podmínky, museli bychom dvakrát opakovat podmínku „pohlavi = F“ nebo „pohlavi = M“. Díky přepínači nemusíme zbytečně opakovat podmínky.

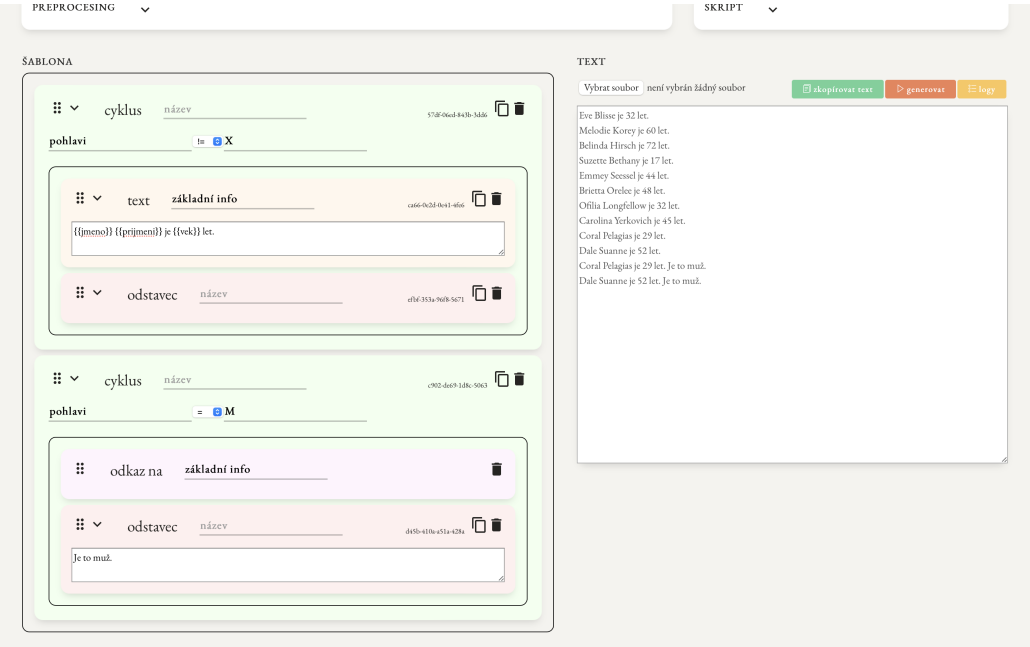


A.1.6 Odkaz

Odkaz může ukazovat na jakýkoli jiný blok pomocí ID nebo názvu daného bloku. Při vyhodnocování se pak odkaz nahradí blokem, na který ukazuje a vyhodnocuje se tento nový blok. Pokud ukazujeme na blok, který využívá konkrétní proměnné, je důležité, aby měl blok k těmto proměnným přístup, pokud tedy například odkazujeme na blok, který je v cyklu a používá jeho proměnné, musí být odkaz také v cyklu a iterovat přes ty samé proměnné.

A.1.6.1 Příklad

Chceme napřed vypsat informace o všech osobách v datech a poté ještě pro všechny muže vypsat další text. V prvním cyklu nadefinujeme podobu základního textu. V druhém cyklu se pak odkážeme na textový blok s názvem „základní info“ a přidáme k němu další text. Díky tomu nemusíme opakovat první část textu. Všimněte si, že podmínka v cyklech sice není stejná, ale cyklus prochází stejná data.

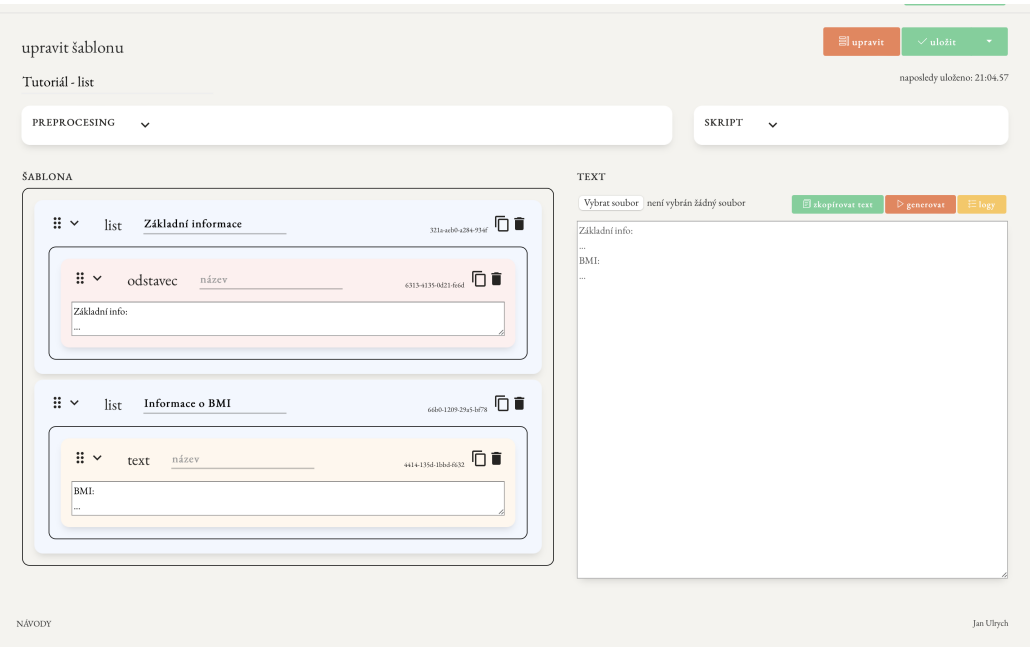


A.1.7 List

List slouží pouze pro vizuální oddělení částí textu.

A.1.7.1 Příklad

Budeme vytvářet delší text. Pro lepší oddělení jednotlivých částí textu použijeme list, který si pro přehlednost můžeme i pojmenovat podle toho, o čem v něm píšeme.



A.1.8 Předzpracování dat

Akce pro předzpracování dat nám pomocí různých způsobů dovolují z dat zjistit různé informace a tyto informace poté uložit do proměnných, které můžeme použít v libovolném místě v šabloně. Akce pro předzpracování dat se definují v tabulce pod názvem šablony. Přidání nové akce provedeme tlačítkem „přidat“. Každá akce musí mít vyplněnou položku „výsledek“, což je název proměnné, do které se uloží výsledná hodnota. Dále musí mít vyplněnou hodnotu „sloupec“, což značí sloupec v datech, nad jehož hodnotami se budou akce provádět. U některých akcí je nutné vyplnit položku „parametr“. Ta je závislá na konkrétní akci.

A.1.8.1 SUM (suma)

Suma provede součet všech hodnot v daném sloupci a výsledek uloží do dané proměnné.

A.1.8.2 AVG (průměr)

Vypočítá průměr všech hodnot v daném sloupci a výsledek uloží do dané proměnné.

A.1.8.3 COUNT (počet)

Spočítá počet výskytů hodnoty dané parametrem v daném sloupci a výsledek uloží do dané proměnné.

A.1.8.4 FIRST (první hodnota)

První hodnotu v daném sloupci uloží do dané proměnné.

A.1.8.5 LAST (poslední hodnota)

Poslední hodnotu v daném sloupci uloží do dané proměnné.

A.1.8.6 MIN (minimum)

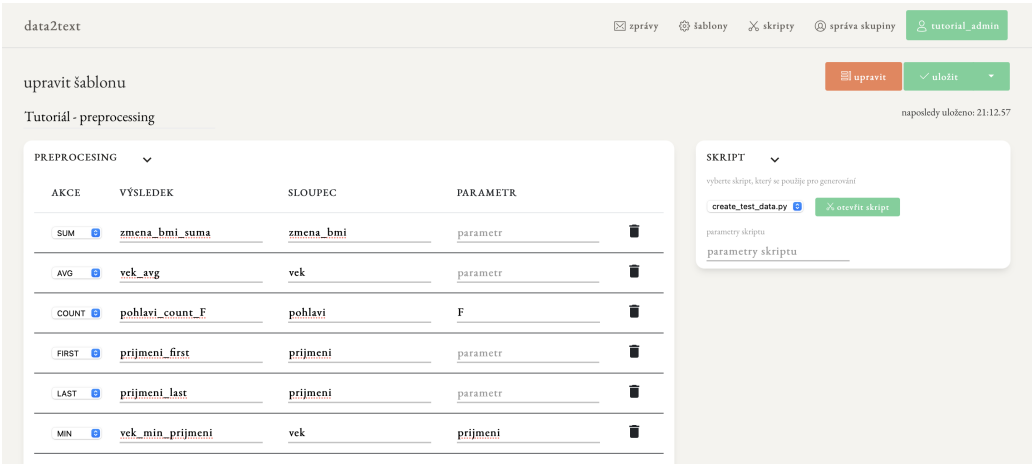
Najde minimum v daném sloupci a jemu odpovídající hodnotu ve sloupci daném parametrem uloží do dané proměnné.

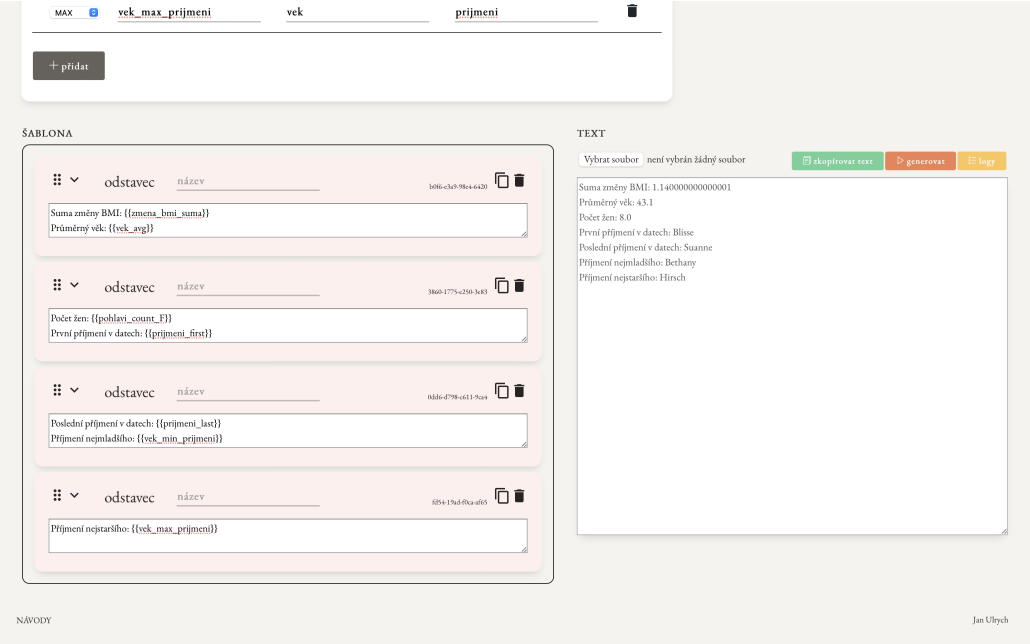
A.1.8.7 MAX (maximum)

Najde maximum v daném sloupci a jemu odpovídající hodnotu ve sloupci daném parametrem uloží do dané proměnné.

A.1.8.8 Příklad

- Sečteme všechny hodnoty ve sloupci „zmena_bmi“ a součet uložíme do „zmena_bmi_suma“.
- Spočteme průměrný věk a uložíme jej do „vek_avg“.
- Použitím akce COUNT najdeme počet všech žen v datech a uložíme jej do „pohlavi_count_F“.
- Najdeme příjmení první osoby v datech.
- Najdeme příjmení poslední osoby v datech.
- Najdeme příjmení nejmladší osoby.
- Najdeme příjmení nejstarší osoby.



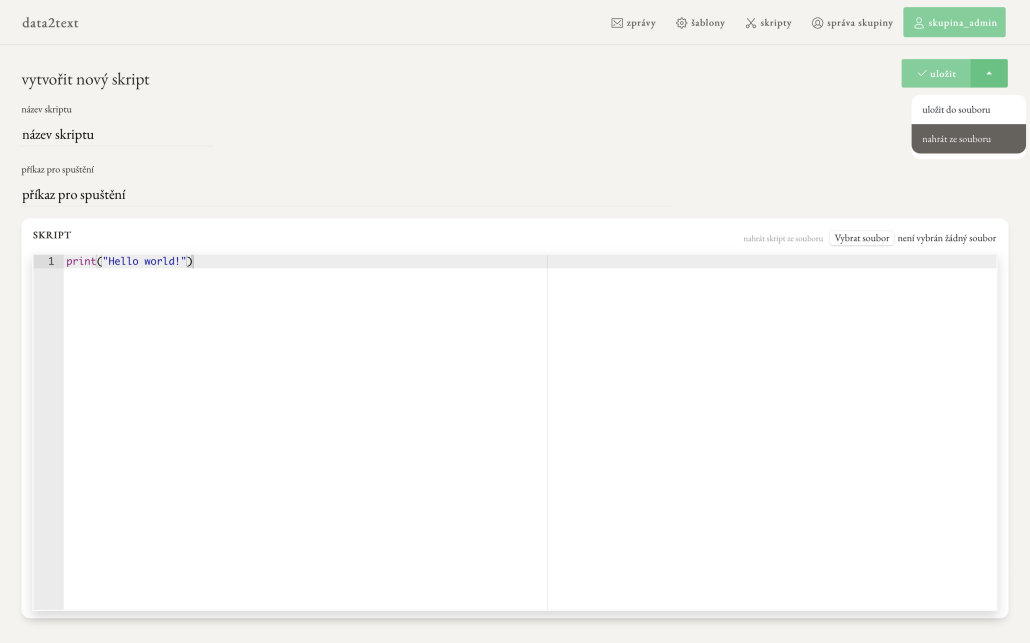


A.2 Skripty

Skripty slouží pro načtení dat z jakéhokoli zdroje, který chce uživatel použít, transformování těchto dat a jejich uložení do podoby, ve které budou dále zpracovány šablonou při generování textu.

A.2.0.1 Ovládací prvky

- Pod nadpisem je možné vyplnit název souboru se skriptem.
- Vpravo nahoře se nachází tlačítko pro uložení skriptu, po rozkliknutí šipky u tlačítka uložit je možné uložit skriptu do souboru nebo jej ze souboru nahrát.
- Pod názvem šablony se nachází pole pro definování příkazu pro spuštění skriptu. Název skriptu v příkazu pro spuštění musí být shodní s názvem skriptu v poli výše. Pokud tedy například spouštíme skript příkazem python skript.py musí být v poli pro název skriptu vyplněno skript.py.
- Pod příkazem pro spuštění se nachází okno pro editaci daného skriptu. V tomto okně můžeme skript rovnou psát, případně do něj skript nakopírovat, nebo ho zde upravit.
- Nad polem pro editaci skriptu se také nachází tlačítko pro nahrání skriptu ze souboru. Pokud nahrajeme skript ze souboru, pole pro editaci skriptu se automaticky vyplní obsahem daného souboru.



A.2.0.2 Výstup skriptu

Výstupem skriptu musí být alespoň jeden soubor ve formátu `.csv`, který je uložen do složky `/output_files` a jeho název začíná předponou `output`. Oddělovačem hodnot v souboru musí být středník (`;`).

A.2.0.3 Testovací data

Pokud při testování šablony nechceme brát data z nějakého vzdáleného zdroje, je možné při generování textu z editoru šablon nahrát testovací soubor. Ten je následně uložen v prostředí skriptu do kořenového adresáře do souboru `data`. Struktura tohoto souboru není nijak specifikována, uživatel může využít jakýkoli testovací soubor, jaký chce.

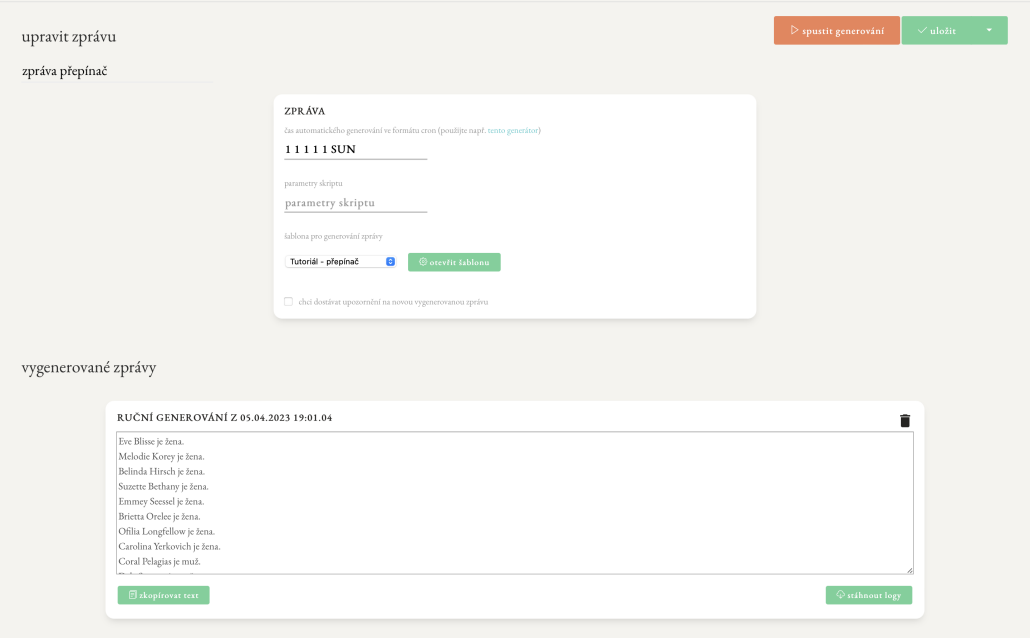
A.3 Zprávy

Zprávy slouží pro automatické generování zpráv z dané šablony a uchování výsledných textů.

A.3.0.1 Ovládací prvky

- Pod nadpisem je možné vyplnit název zprávy.

- Vpravo nahoře se nachází tlačítko pro uložení skriptu, po rozkliknutí šipky u tlačítka uložit je možné uložit skriptu do souboru nebo jej ze souboru nahrát.
- Vedle tlačítka pro uložení skriptu je tlačítko pro manuální generování zprávy.
- V prvním poli na bílém pozadí je možné vyplnit čas automatického generování ve formátu SpringBoot cron. Pro vytvoření tohoto textového řetězce je možné využít generátor odkazovaný nad polem pro zadání tohoto řetězce.
- Pod časem automatického generování je možné vyplnit parametry skriptu, tyto parametry budou přidány na konec příkazu pro spuštění skriptu.
- Pod parametry skriptu je nutno vybrat šablonu, podle které se budou zprávy generovat. Také je možné šablonu rovnou otevřít.
- Zaškrtávacím tlačítkem můžeme určit, zda chceme dostávat upozornění na novou vygenerovanou zprávu pomocí emailu.



A.3.0.2 Přehled vygenerovaných zpráv

Pod nastavením zprávy se nachází přehled všech vygenerovaných zpráv. Pokud se zprávu nepodařilo vygenerovat, okno bude červeně podbarvené a uživatel zde uvidí souhrnný popis chyby. Detailní informace poté zjistí z logů.

- Vygenerovanou zprávu je možné odstranit tlačítkem se symbolem odpadkového koše.
- Stisknutím tlačítka „zkopírovat text“ se vygenerovaný text nakopíruje do schránky.
- Po kliknutí na tlačítko „stáhnout logy“ bude stažen soubor ve formátu zip obsahující soubory použité v rámci generování dané zprávy.

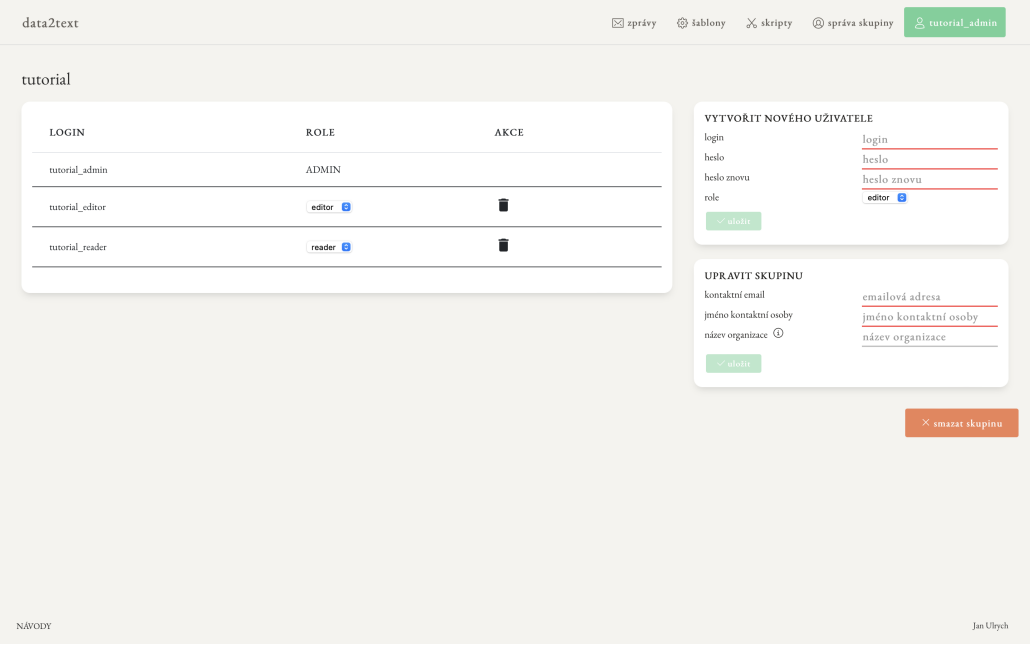
A.4 Administrace skupiny

Uživatelé s rolí Admin mají přístup k administraci uživatelů ve své skupině. V záložce „správa skupiny“ mohou vidět seznam všech uživatelů registrovaných pod jejich skupinou, mohou vytvořit nového uživatele, upravit detaily skupiny a také skupinu smazat.

V levé části obrazovky se nachází přehled všech uživatelů. Tabulka obsahuje přihlašovací jméno daného uživatele, prostřední sloupec pak ukazuje aktuální roli uživatele, která může být výběrem jiné role změněna, v posledním sloupci je pak možnost daného uživatele, po kliknutí na ikonu odpadkového koše, smazat.

V pravé horní části obrazovky je pak možné přidat nového uživatele. Uživateli je možné nastavit přihlašovací jméno, heslo a roli.

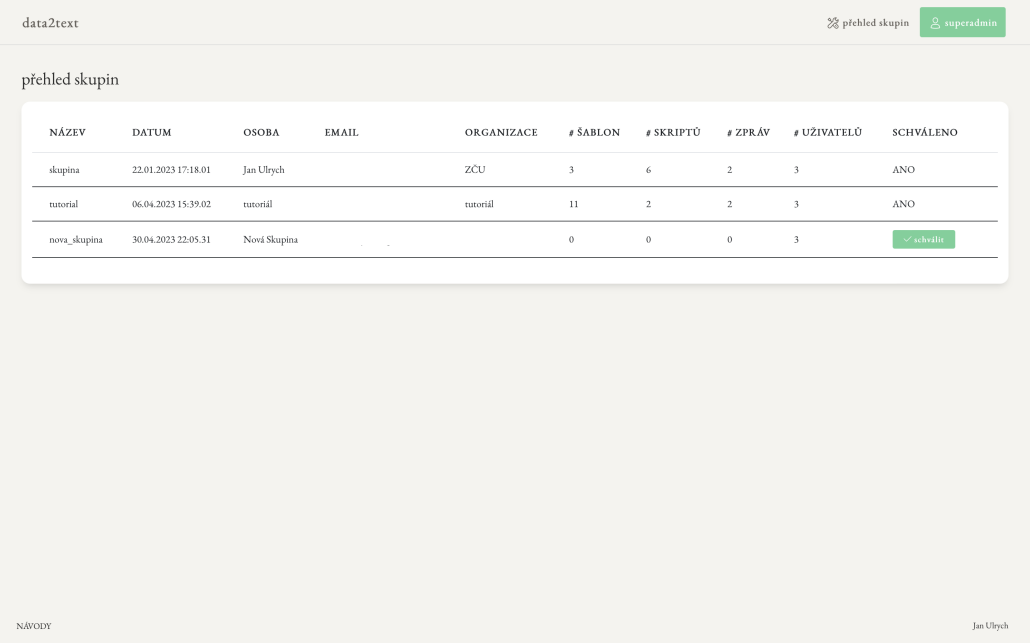
V pravé dolní části obrazovky je pak možné upravit detaily skupiny - emailovou adresu na vlastníka skupiny, jméno kontaktní osoby a případně název organizace. Po kliknutí na tlačítko „smazat skupinu“ bude skupina se všemi jejími objekty (zprávami, šablonami a skripty) smazána.



A.5 Administrace systému

Uživatel s rolí Superadmin má právo vidět přehled všech skupin. V přehledu všech skupin jsou obsaženy informace o názvu skupiny, datum registrace dané skupiny, osoba, která danou skupinu vytvořila, kontaktní emailová adresa, organizace, jíž je daná skupina součástí, počet šablon, skriptů, zpráv a uživatelů a možnost danou skupinu schválit (kliknutím na zelené tlačítko „schválit“), pokud ještě schválena nebyla.

Po registraci nové skupiny přijde Superadminovi na jeho emailovou adresu upozornění na nově vytvořenou skupinu. Po schválení dané skupiny je pak na uvedený kontaktní email odeslána informace o schválení skupiny.



Burzovní zpravodajství

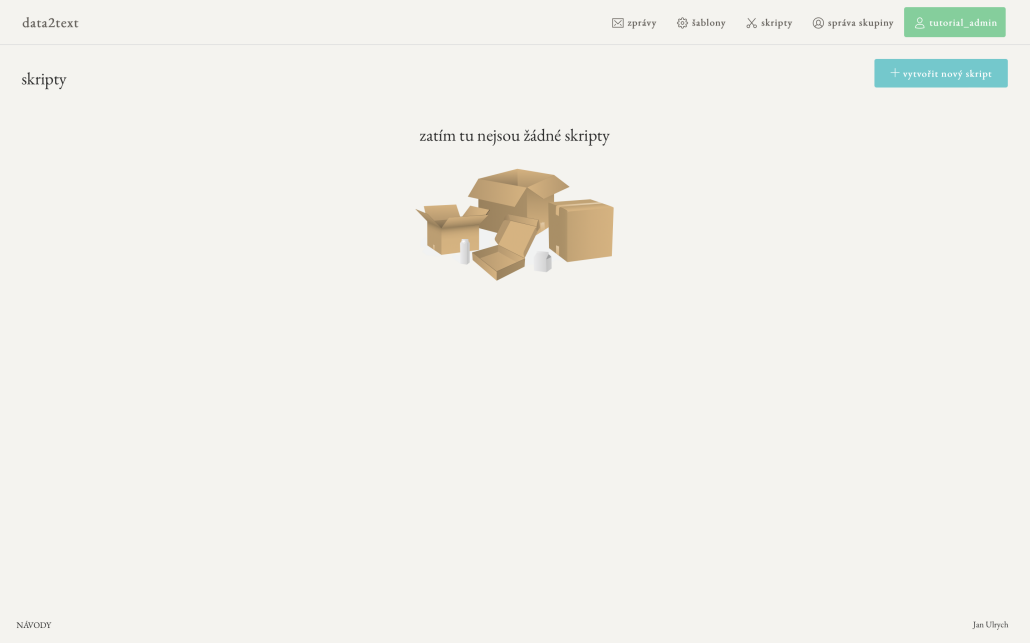
B

B.1 Skript

Pro vygenerování textu pojednávajícím o pohybech pražské burzy budeme potřebovat skript, který stáhne data z FTP serveru burzy, tato data transformuje a uloží do výstupních souborů. Postup vytvoření nového skriptu je následující:

B.1.0.1 1. krok:

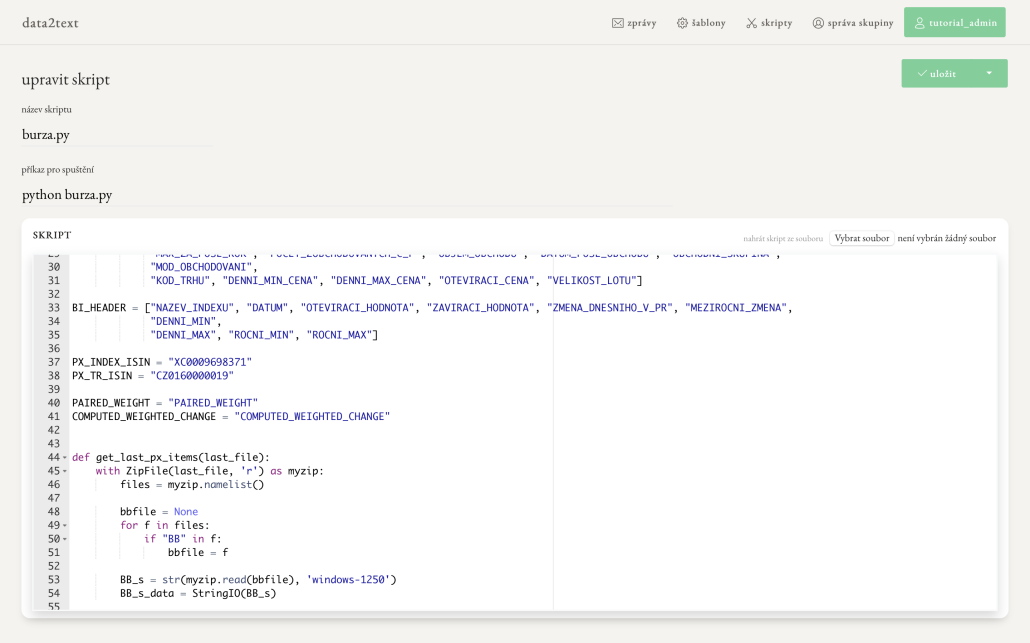
V záložce skripty klikneme na tlačítko „vytvořit nový skript“.



B.1.0.2 2. krok:

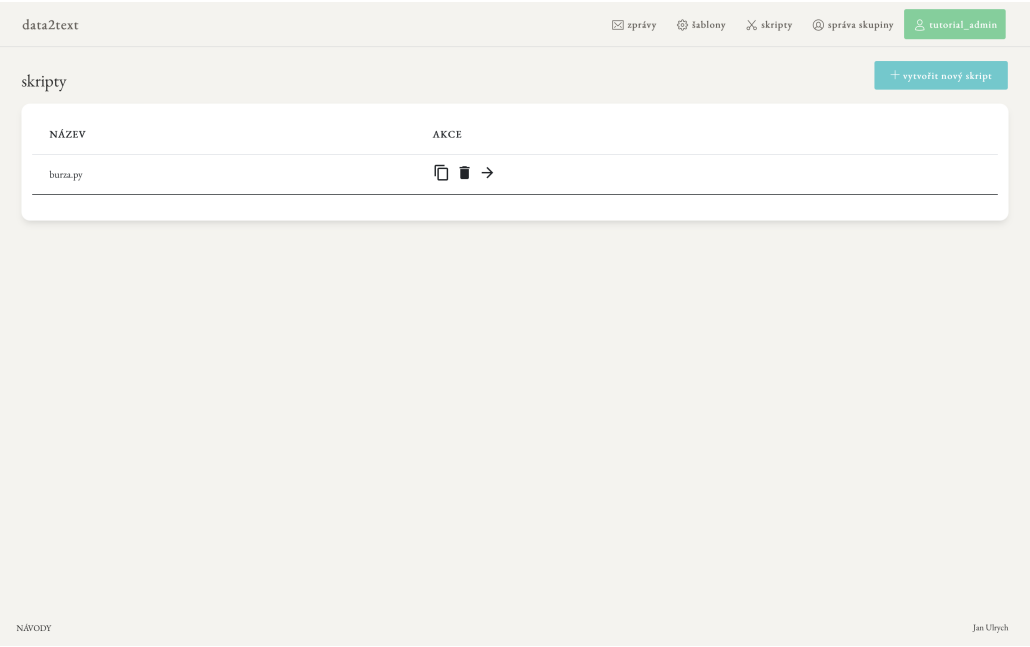
Vyplníme údaje „název skriptu“, „příkaz pro spuštění“ a dále samotný skript, který můžeme do pole buď vložit po předchozím zkopírování z jiného zdroje, nebo vložit

nahráním souboru. Skript je přílohou této práce. Po vyplnění těchto údajů skript uložíme kliknutím na tlačítko „uložit“.



B.1.0.3 3. krok:

Po úspěšném uložení skriptu budeme přesměrování na přehled všech skriptů, kde můžeme vidět náš nově vytvořený skript.



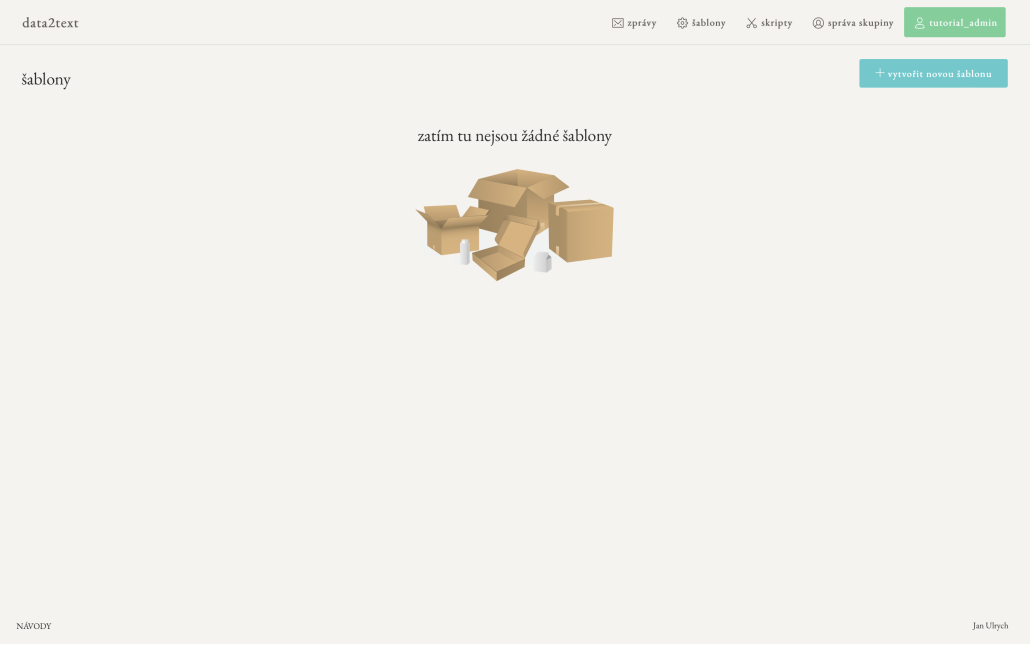
To je vše, co pro vytvoření nového skriptu budeme potřebovat. Tento skript nyní můžeme používat v šablonách. Pokud bychom chtěli generovat text pro včerejší data, můžeme skriptu předat parametr 1 (pro dva dny do minulosti 2 atd.).

B.2 Šablona

To, jak bude výsledný text vypadat, definujeme pomocí šablony. Při generování textu se nejdříve vykoná definovaný skript, který vytvoří výstupní soubory s daty pro šablonu. Tato data se poté zpracují, vyhodnotí se akce pro předzpracování dat a na závěr se nad danými daty vyhodnotí šablona. Vytvoření šablony je obdobné, jako vytváření skriptů.

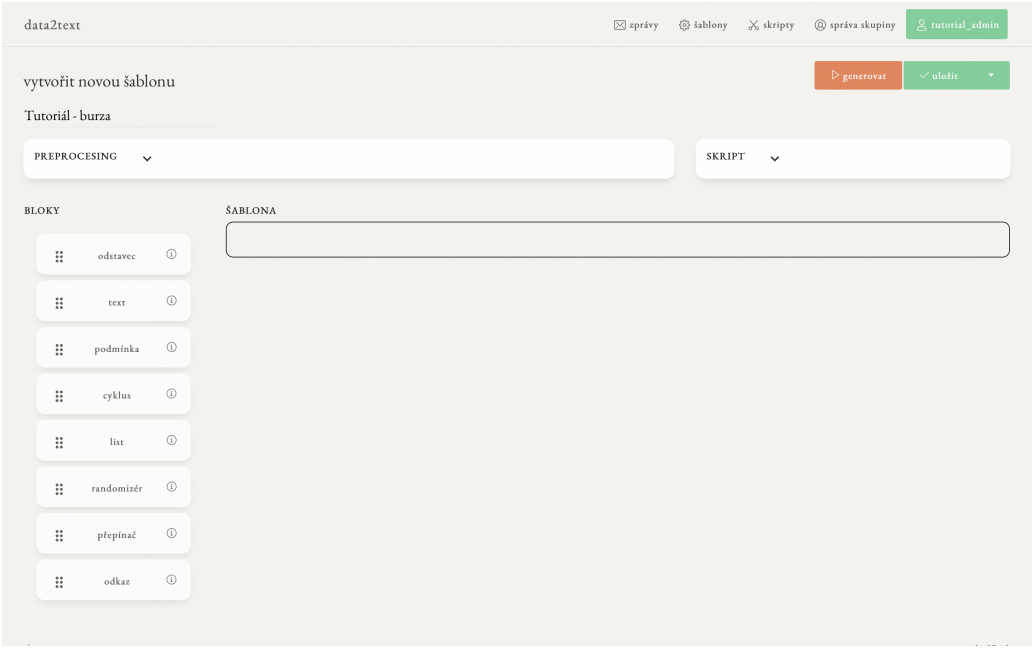
B.2.0.1 1. krok:

V záložce šablony klikneme na tlačítko „vytvořit novou šablonu“.



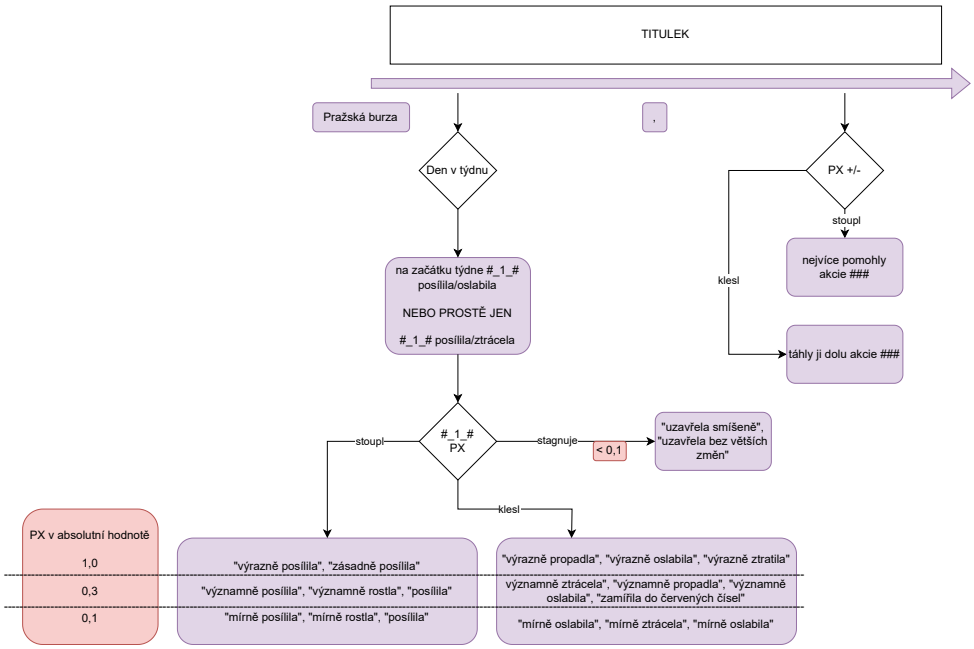
B.2.0.2 2. krok:

Vyplníme název šablony a šablonu uložíme.



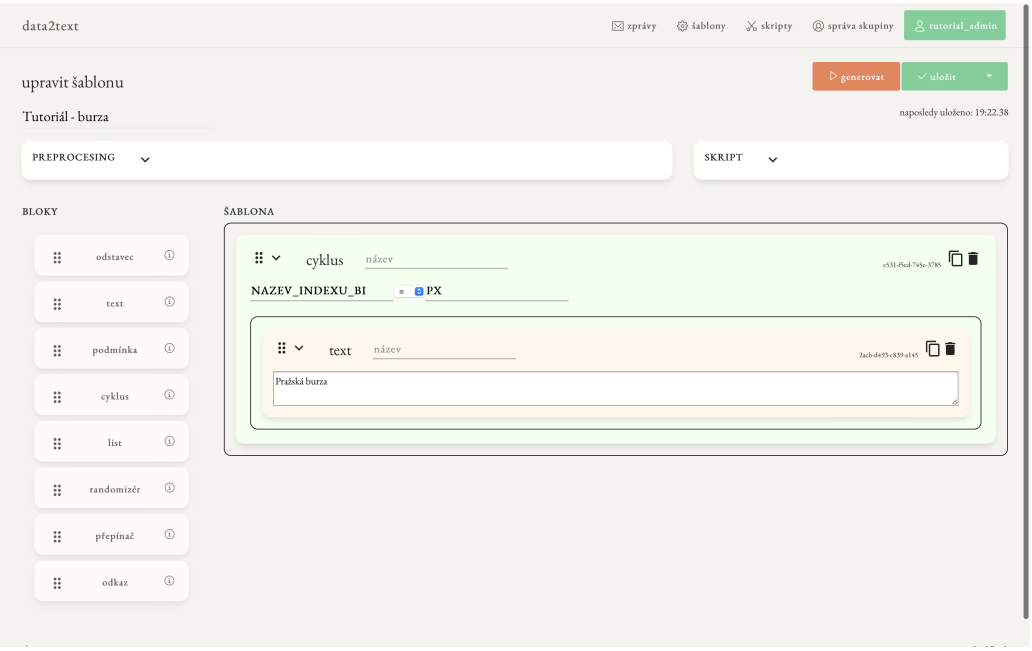
B.2.1 Titulek

Jako první budeme definovat šablonu pro titulek. Dosavadní systém generuje titulek podle následujícího diagramu:



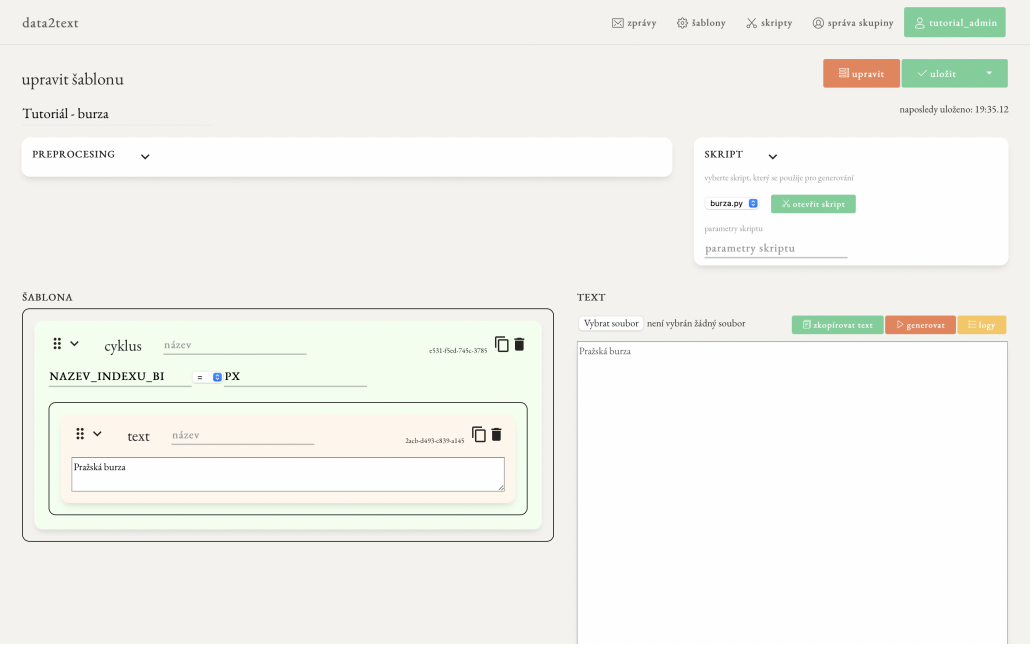
B.2.1.1 1. krok:

Informace o pražské burze najdeme v souboru „output_bi.csv“. V řádce, která má ve sloupci „NAZEV_INDEXU_BI“ hodnotu „PX“ pak nalezneme informace o indexu PX, který budeme popisovat. Cyklem tedy projdeme všechny hodnoty ve sloupci „NAZEV_INDEXU_BI“ a bloky budeme vykonávat pouze v případě, že nalezneme hodnotu „PX“. Ze sloupce bloky přetáhneme do sloupce šablona blok s názvem „cyklus“, nastavím levou stranu na hodnotu „NAZEV_INDEXU_BI“, operátor na „=“ a pravou stranu na hodnotu „PX“. Do cyklu můžeme také vložit text „Pražská burza“, jelikož jim bude začínat každý titulek. Nezapomeňte také za text „Pražská burza“ dát mezeru.



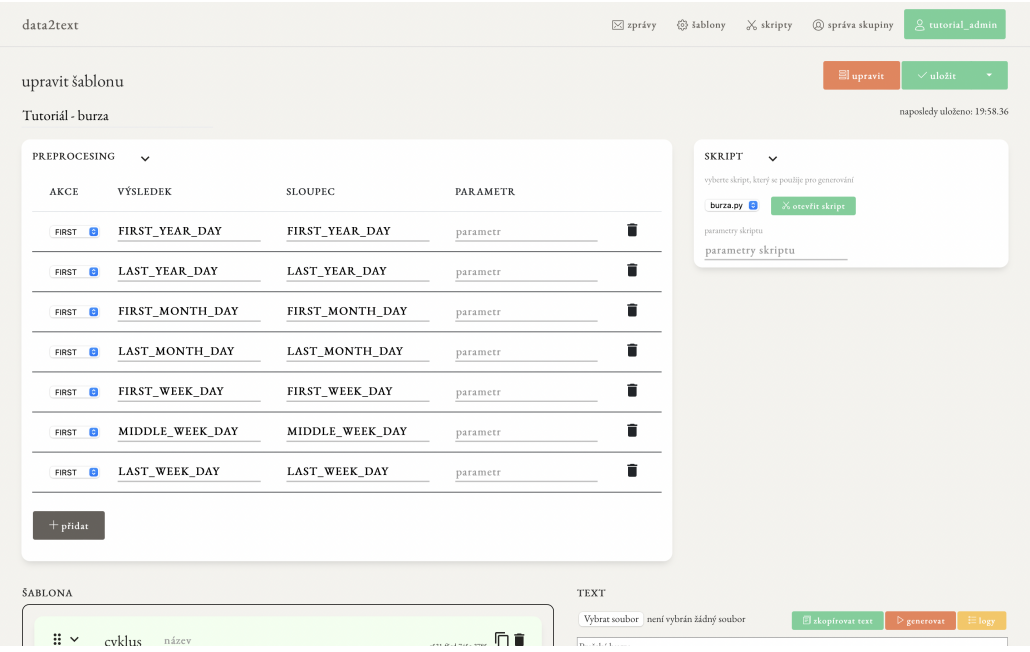
B.2.1.2 2. krok:

Nyní by se měl vygenerovat text „Pražská burza“. Před otestováním ještě zkontrolujeme, že máme v tabulce „Skript“ zvolen skript „burza.py“. Samotné otestování provedeme kliknutím na tlačítko „generovat“, poté se zobrazí další tlačítko „generovat“, na které klikneme a počkáme, než se vygeneruje text. Pokud je vše správně, zobrazí se text „Pražská burza“ (pouze jednou, protože v sloupci „NAZEV_INDEXU_BI“ je pouze jedna hodnota „PX“).



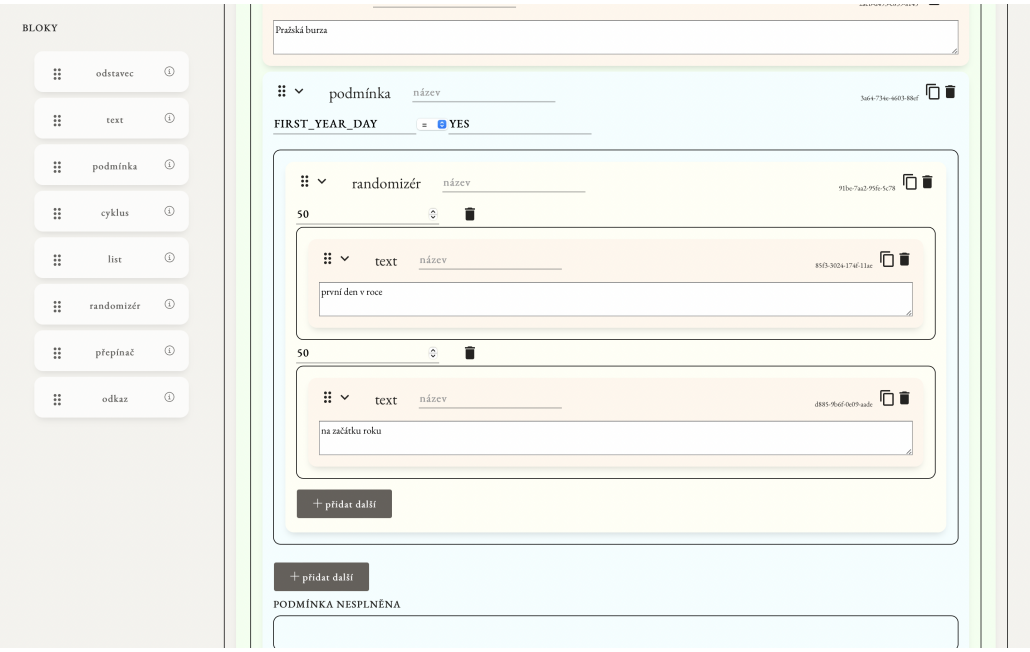
B.2.1.3 3. krok:

Podle diagramu se v další části budeme zabývat tím, zda je nyní nějaký „významný den“. Data o tom jsou generována do souboru „output_days.csv“. Abychom se k těmto datům dostali, musíme nadefinovat akce pro předzpracování dat. Pro tento účel využijeme akci „FIRST“, ze sloupečku „FIRST_YEAR_DAY“ vezmeme hodnotu a uložíme ji do pole „FIRST_YEAR_DAY“. Stejně tak pro další významné dny. Tabulka akcí pro předzpracování dat pak bude vypadat následovně:



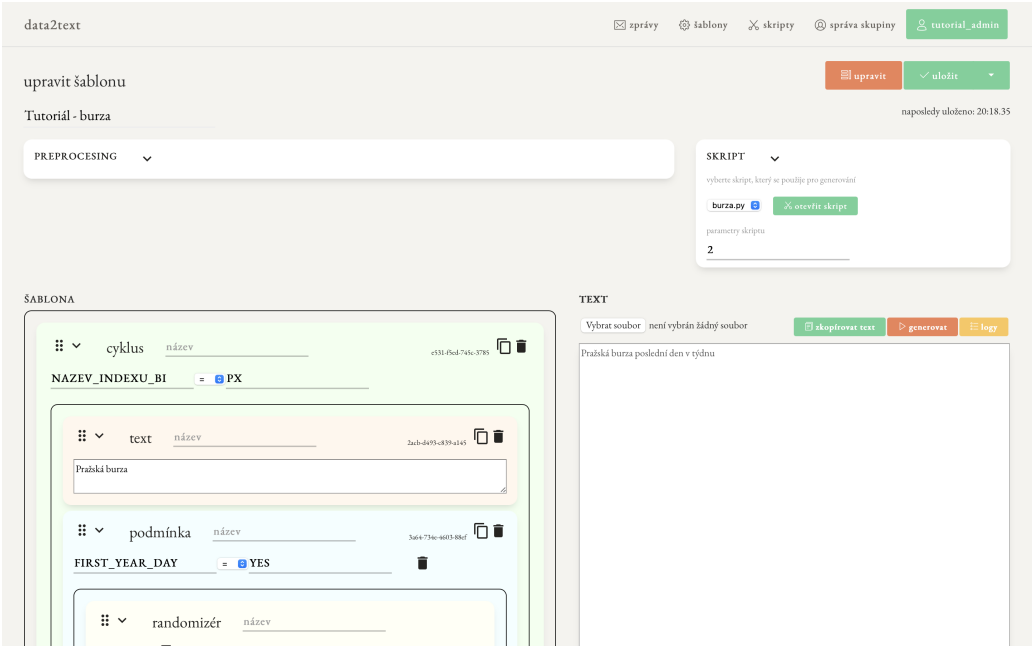
B.2.1.4 4. krok:

Poté do cyklu přidáme podmínku a budeme kontrolovat, zda se nějaké z těchto nově nadefinovaných polí rovná hodnotě „YES“ a pokud ano, pomocí randomizéru vypíšeme odpovídající text. Např. pro první den v roce bude podmínka vypadat následovně:



B.2.1.5 5. krok:

Následně doplníme i ostatní podmínky a můžeme otestovat generování. Pokud zrovna dnešek není „významný den“, můžete pomocí parametrů skriptu v tabulce „Skript“ zadat počet dní (kladné celé číslo), které vás posune o zadaný počet dní do minulosti.



B.2.1.6 6. krok:

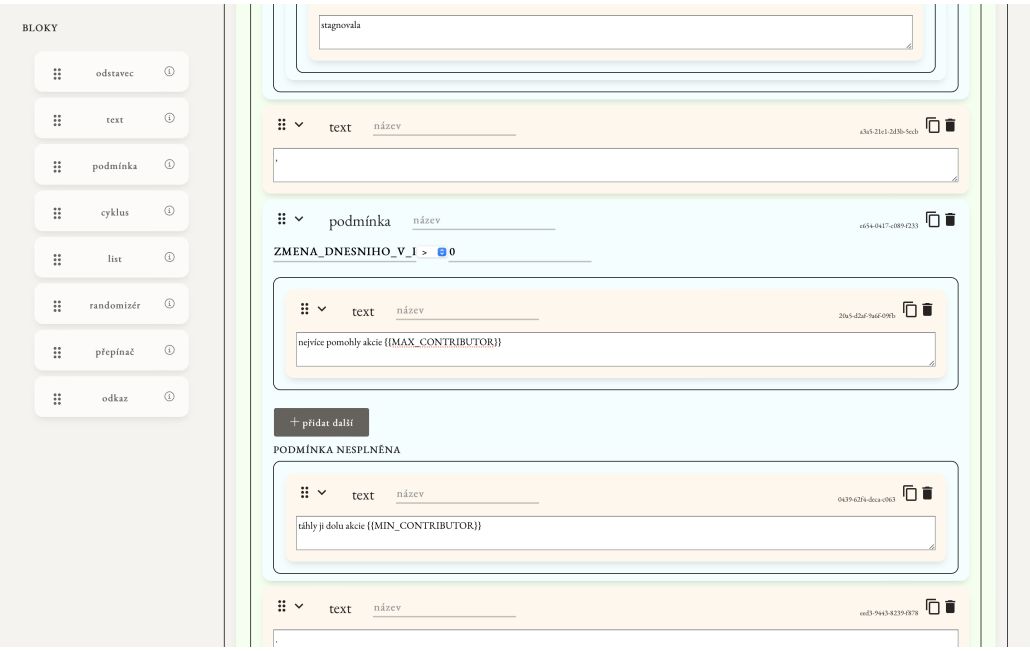
Nyní se v diagramu posuneme na informaci o tom, jak se pražské burze dařilo. Pro určení této informace budeme potřebovat hodnotu ve sloupci „ZMENA_DNESNIHO_V_PR_BI“. Na základě této hodnoty se budeme rozhodovat, jaký text vygenerujeme. Použijeme tedy opět podmínku s textem. Texty poté můžeme i randomizovat. Podmínka pro rostoucí burzu bude vypadat takto:

B.2.1.7 7. krok:

Na konec titulku přidáme informaci o nejúspěšnější či nejméně úspěšné akcii (na základě pohybu burzy). Pro tento účel použijeme akce pro předzpracování dat „MIN“ a „MAX“. Hledáme minimální hodnotu sloupce „COMPUTED_WEIGHTED_CHANGE_AK“ a do pole „MIN_CONTRIBUTOR“ si uložíme název akce ze sloupce „NAZEV_AK“.

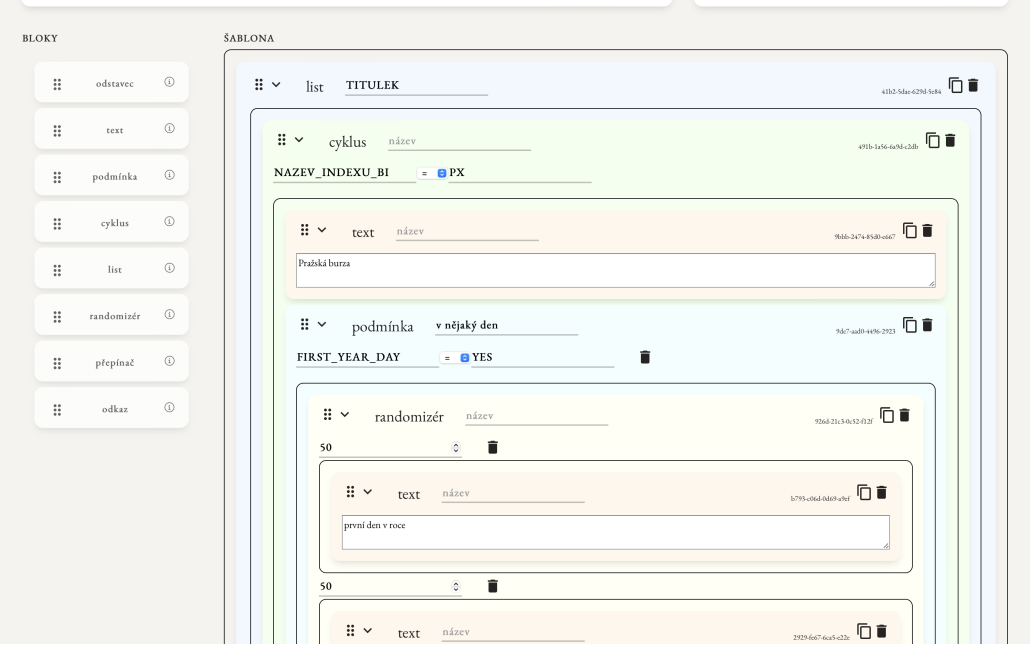
B.2.1.8 8. krok:

Do šablony pak přidáme podmínky, na základě které se rozhodneme, zda burza klesala, či rostla a podle toho vypíšeme buď název nejúspěšnější akcie, nebo té nejméně úspěšné.



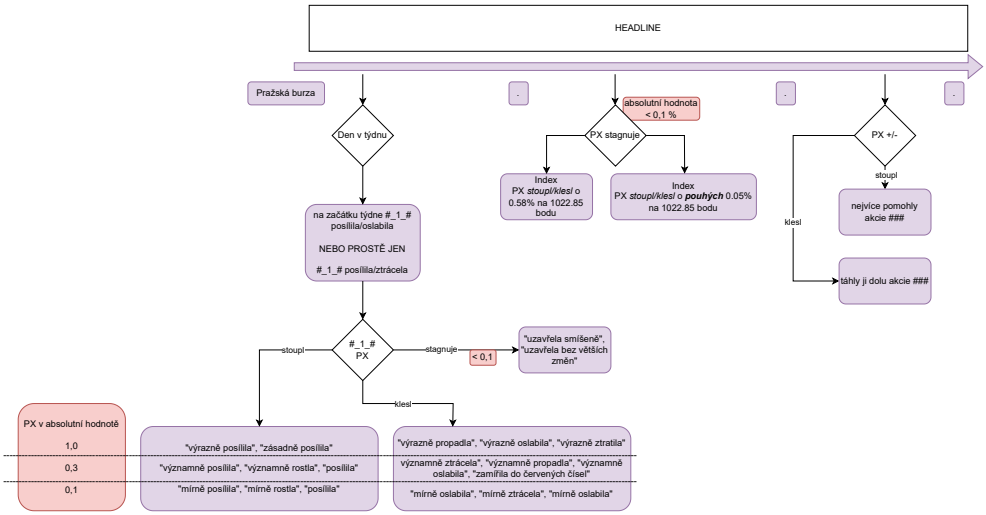
B.2.1.9 9. krok:

Výsledná zpráva by pak měla být zhruba taková: „Pražská burza poslední den v týdnu výrazně oslabila, táhly ji dolu akcie ERSTE GROUP BANK“. Pro přehlednost ještě všechny bloky vložíme do listu, který pojmenujeme jako „TITULEK“.



B.2.2 Headline

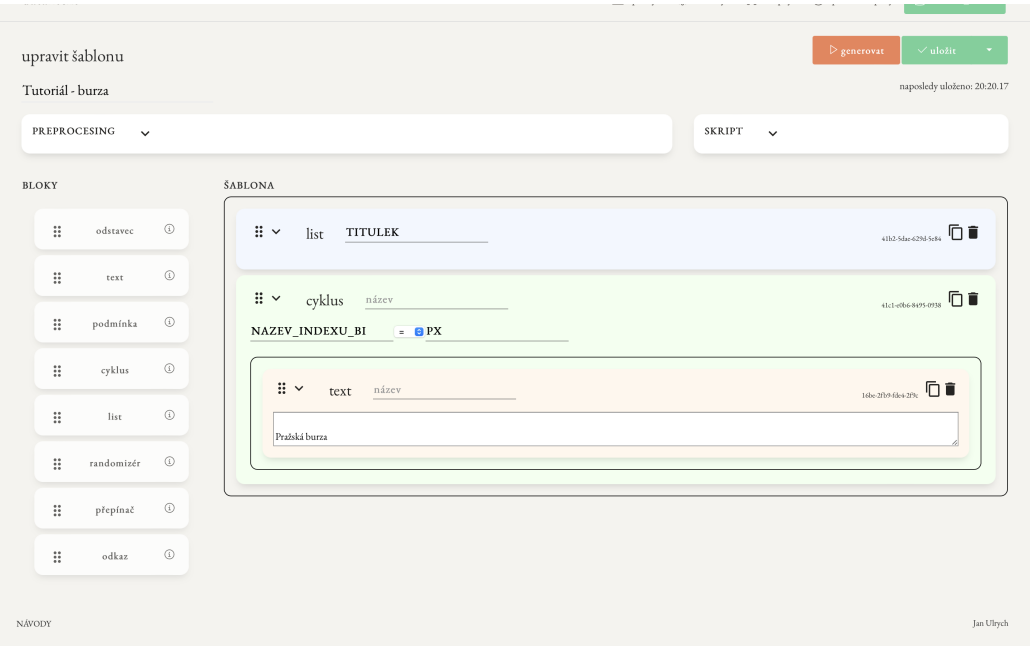
Nyní budeme pokračovat v implementaci šablony pro headline, což je titulek rozšířený o několik informací. Stávající systém generuje headline podle následujícího diagramu:



B.2.2.1 1. krok:

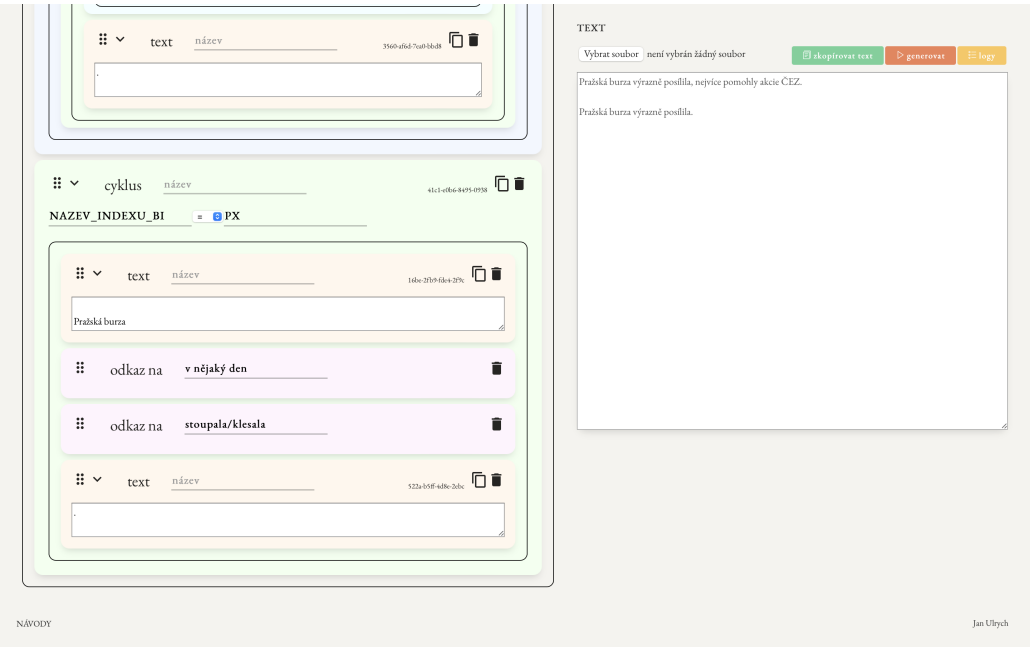
Jelikož opět budeme hledat informace o burze, budeme vycházet z dat ze souboru „output_bi.csv“. Konkrétně budeme opět v cyklu procházet sloupec „NAZEV_INDE-

XU_BI“ a pokud se jeho hodnota bude rovnat hodnotě „PX“, vyhodnotíme bloky. Přidáme tedy cyklus, ve kterém se bude vypisovat text „Pražská burza “.



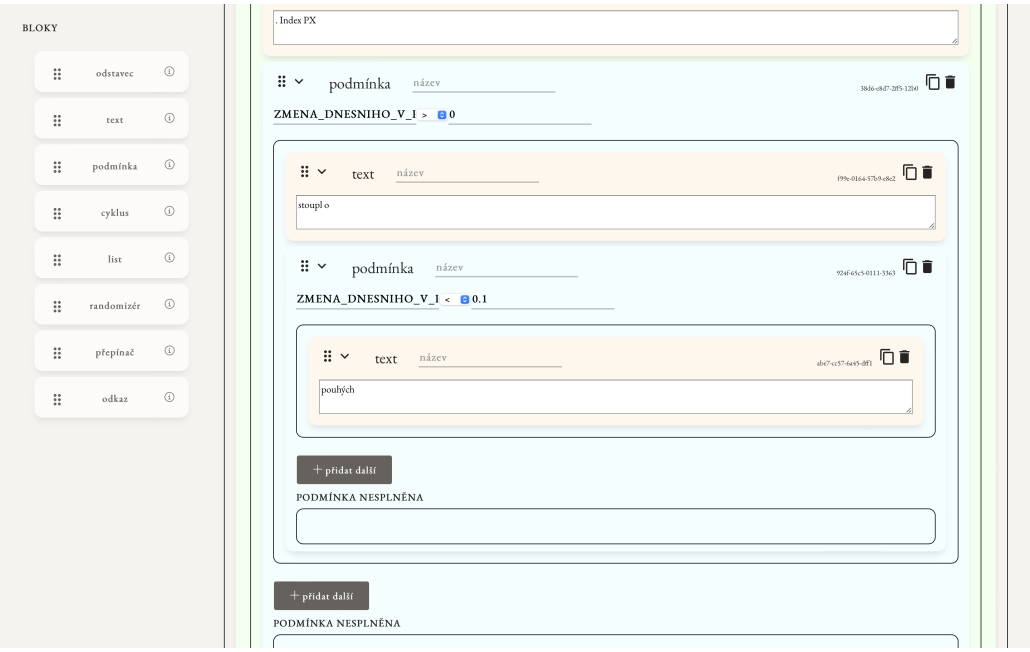
B.2.2.2 2. krok:

Následně vypíšeme informaci o významném dni a informaci o tom, jestli burza stoupala, či klesala. Jelikož jsme šablonu pro výpis těchto informací již implementovali, využijeme blok „odkaz“, který bude odkazovat na „titulkovou“ podmínku určující významný den (kde podmínka první větve je „FIRST_YEAR_DAY = YES“), tu si pojmenujeme například „v nějaký den“ (musíme se tedy vrátit v šabloně výše a pole název u podmínky vyplnit). Dále budeme odkazovat na podmínku vypisující informace o pohybu burzy, tu si stejně jako v předchozím případě pojmenujeme, nyní například „stoupala/klesala“. Poté již můžeme do cyklu vložit dva odkazy s patřičnými hodnotami. Šablona i s vygenerovaným textem bude vypadat takto:



B.2.2.3 3. krok:

Následovat bude oproti titulku bude přesnější informace o pohybu indexu. Budeme vypisovat informaci o tom, jestli index klesl či stoupl a následně i přesnou hodnotu indexu. Pokud tedy hodnota sloupce „ZMENA_DNESNIHO_V_PR_BI“ bude kladná, vypíšeme, že stoupl. Pokud ale stoupl o méně než 0.1 %, vložíme do textu slovo „pouhých“:

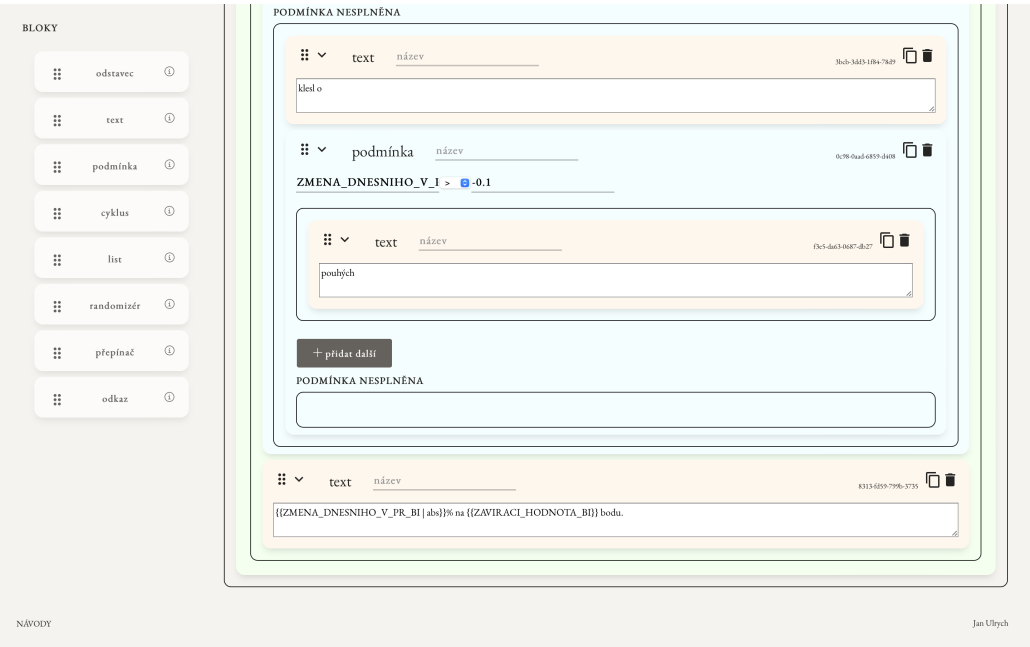


B.2.2.4 4. krok:

Analogicky i pro podmínku nesplněnou, tedy změna indexu je záporná, tudíž index klesl:

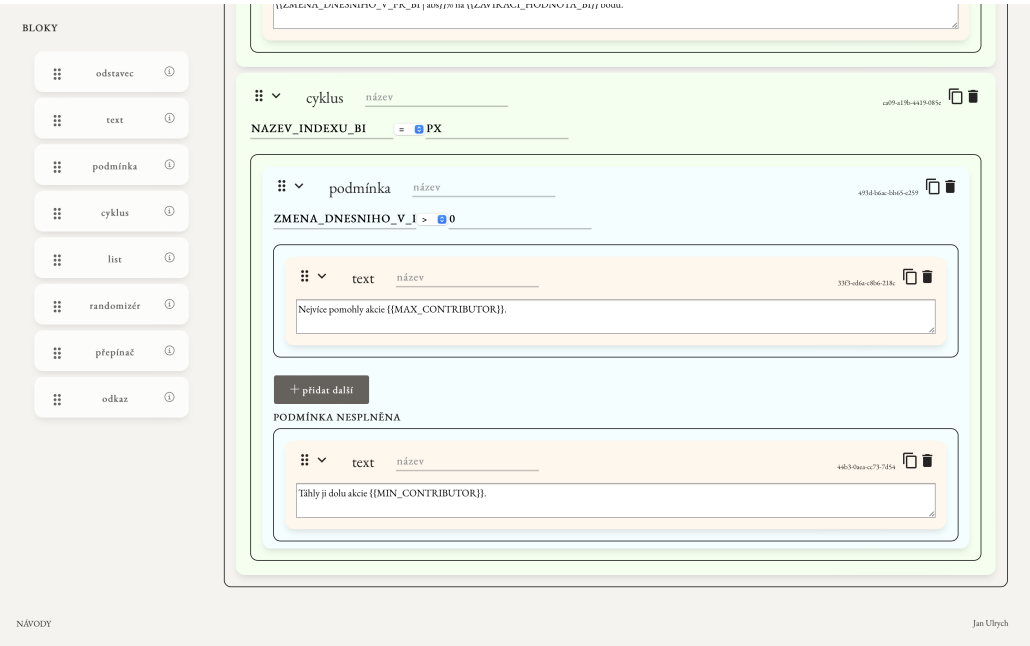
B.2.2.5 5. krok:

Následuje informace o konkrétních hodnotách indexu, tedy změně v absolutní hodnotě a o zavírací hodnotě indexu:



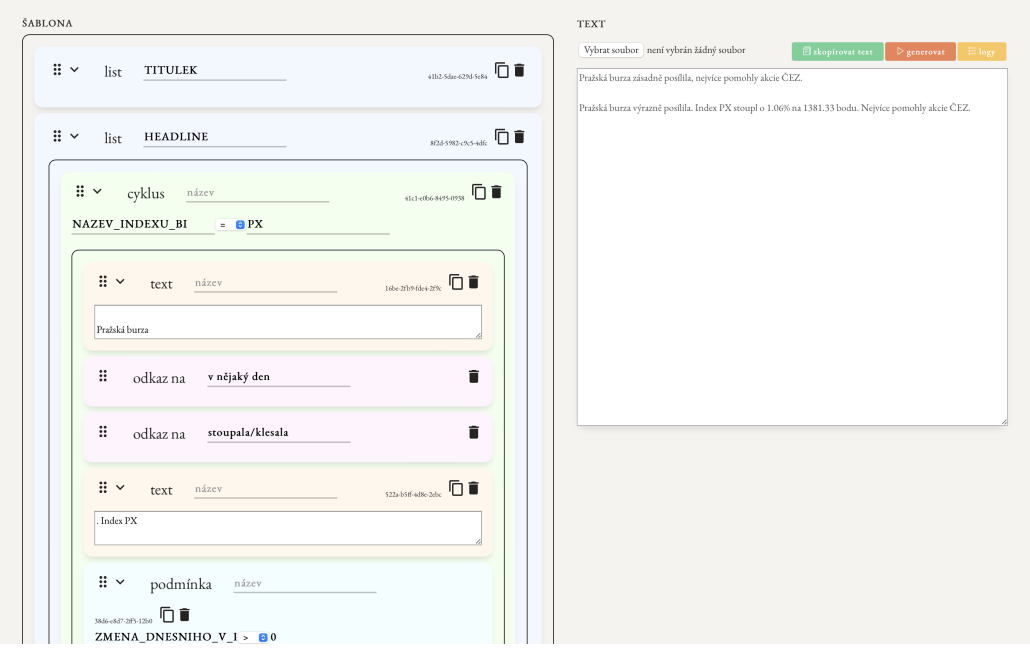
B.2.2.6 6. krok:

Jelikož předchozí informace budeme později v textu ještě jednou opakovat, pro následující část nadefinujeme nový cyklus, ale se stejnou podmínkou jako předchozí. Budeme vypisovat informace o tom, jaké akcie indexu nejvíce pomohly, či ho nejvíce táhly dolu, k tomu využijeme výsledky již dříve definovaných akcí pro předzpracování dat „MAX_CONTRIBUTOR“ a „MIN_CONTRIBUTOR“. Opět budeme větvit na základě změny hodnoty indexu, tedy hodnoty sloupce „ZMENA_DNESNIHO_V_PR_BI“:



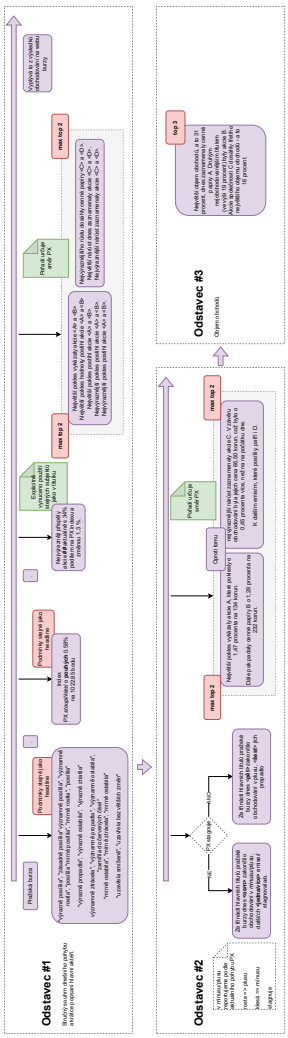
B.2.2.7 7. krok:

Nakonec ještě pro přehlednost všechny nové bloky vložíme do listu s názvem „HEADLINE“ a vygenerujeme si text:



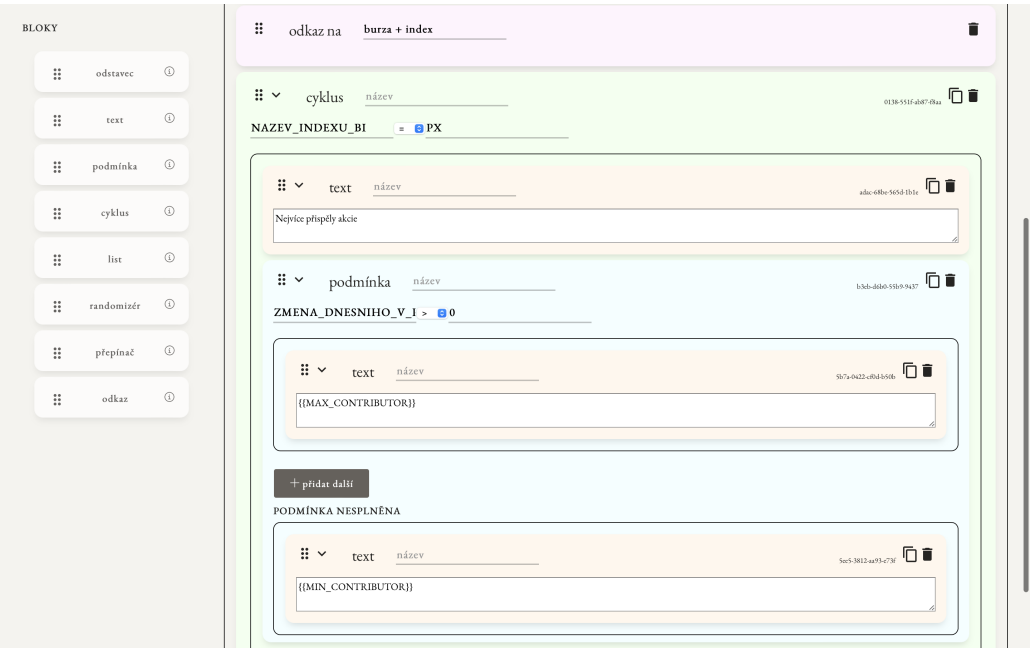
B.2.3 Tělo

Nyní budeme definovat šablonu pro samotné tělo zprávy, které je ve stávajícím systému generováno podle následujícího diagramu:



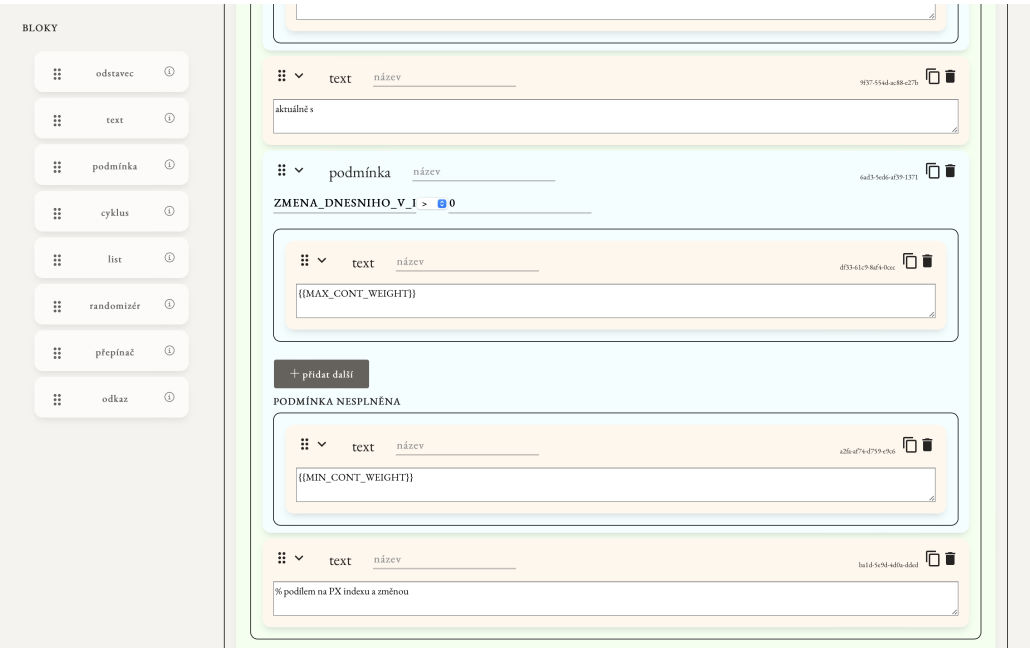
B.2.3.1 1. krok:

V prvním odstavci budeme vypisovat informace obsažené v headline rozšířené o dodatečná data. Odkážeme se tedy na první cyklus v listu „HEADLINE“, který si pojmenujeme jako „burza + index“. Dále budeme opět v cyklu procházet sloupec „NAZEV_INDEXU_BI“ souboru „output_bi.csv“ a bloky vyhodnotíme pouze pokud se hodnota tohoto sloupce rovná hodnotě „PX“. Následně vypíšeme informace o největších přispěvatelích do indexu:



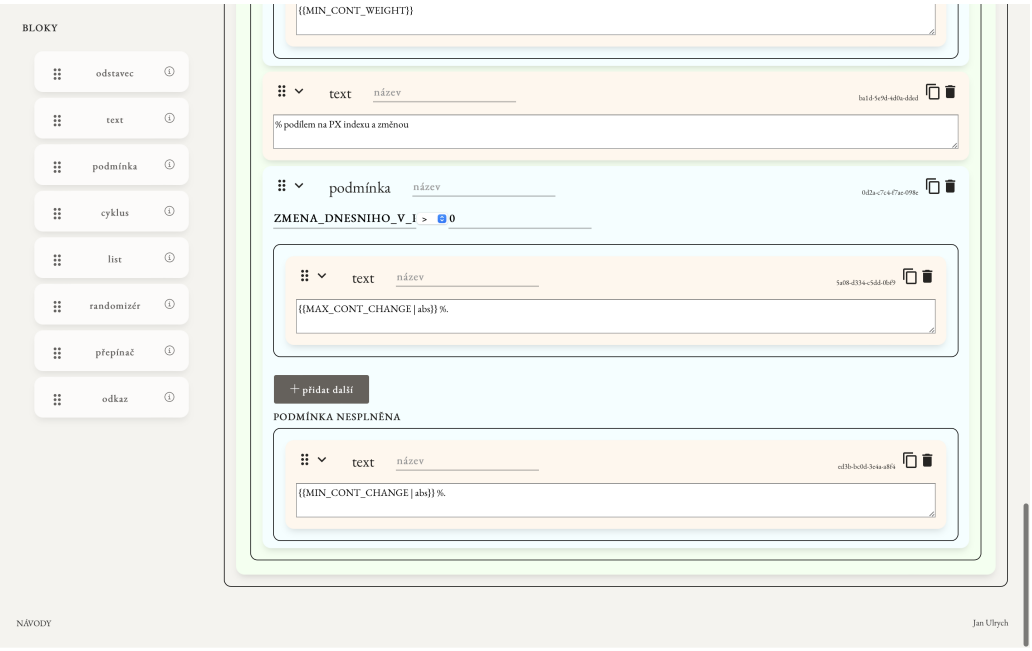
B.2.3.2 2. krok:

Abychom informace oproti headline rozšířili, přidáme data o váze daných přispěvatelů v poměru k celé burze. K tomu nadefinujeme akci pro předzpracování dat „MAX“ s výsledkem „MAX_CONT_WEIGHT“ pro sloupec „ZMENA_DNESNIHO-_V_PR_AK“ a parametrem „PAIRED_WEIGHT_AK“. Obdobně pro minimum. Šablona pak bude vypadat následovně:



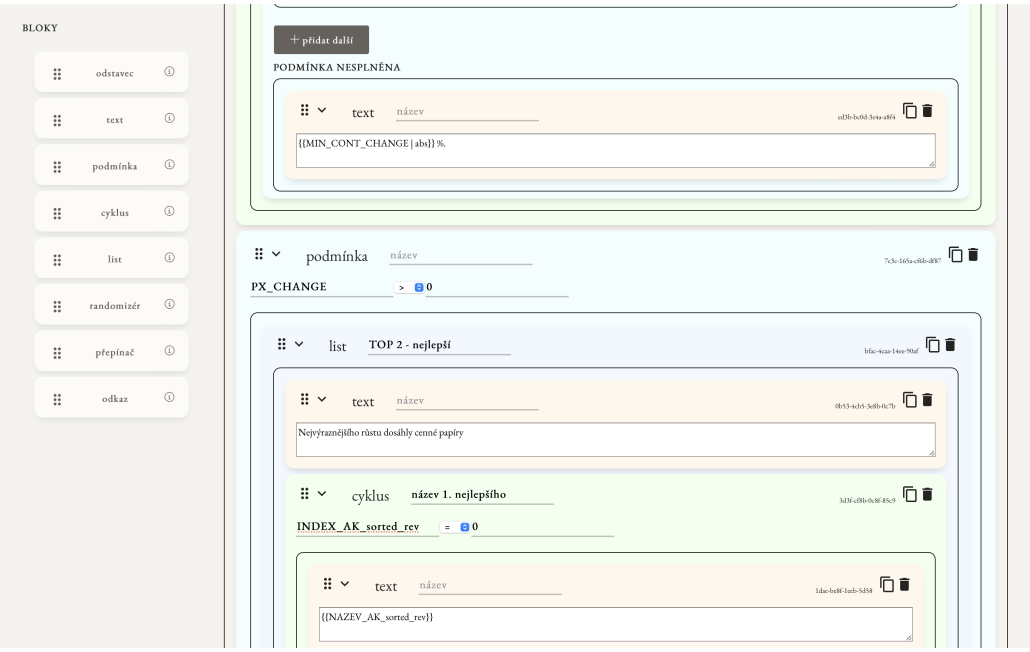
B.2.3.3 3. krok:

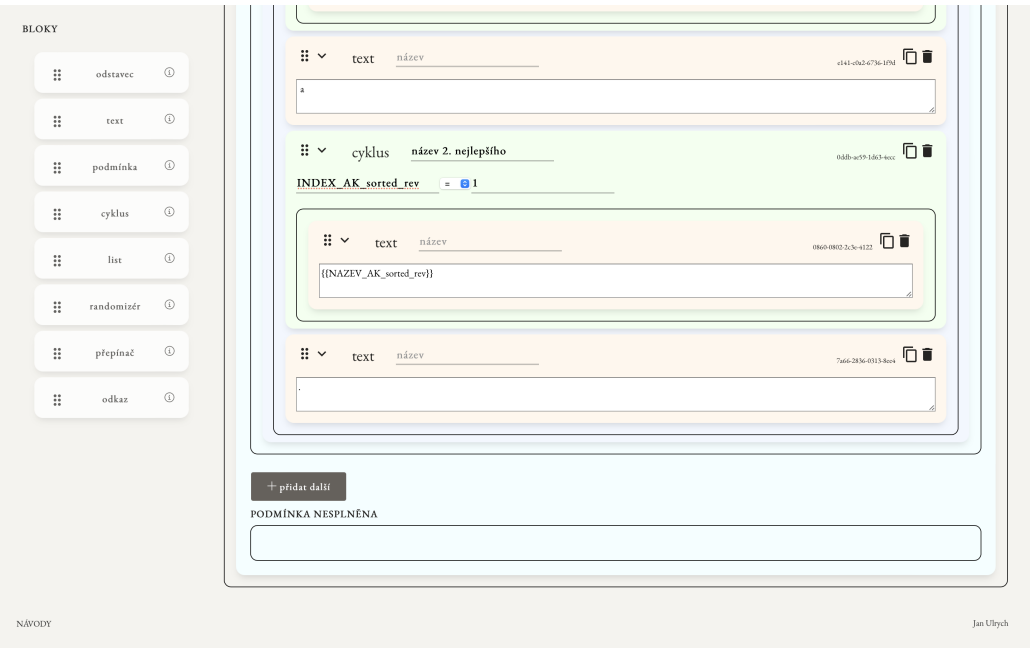
Pro změnu přispěvatelů si nadefinujeme další akci pro předzpracování dat „MAX“ s výsledkem „MAX_CONT_CHANGE“ pro sloupec „ZMENA_DNESNIHO_V_-PR_AK“ a s parametrem „ZMENA_DNESNIHO_V_PR_AK“. Obdobně pro minimum. Výsledky těchto akcí pak vypíšeme v absolutní hodnotě:



B.2.3.4 4. krok:

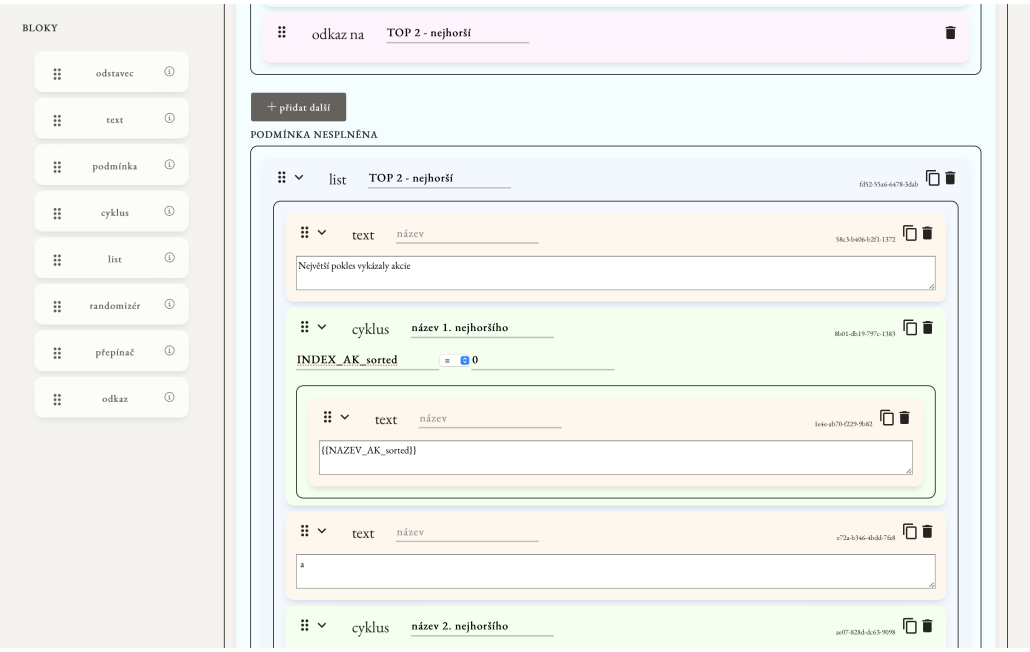
Následovat bude informace o dvou nejúspěšnějších a dvou nejneúspěšnějších akciích. Abychom nemuseli stále procházet data v cyklu, nadefinujeme si akci pro předzpracování dat „AVG“ s výsledkem „PX_CHANGE“ pro sloupec „ZMENA_DNESNIHO-_V_PR_BI“. Ta vypočítá průměr hodnot ve sloupci „ZMENA_DNESNIHO_V_PR-_BI“, který použijeme jako ukazatel, zda burza klesala, či stoupala. Pokud burza stoupala, vypíšeme nejprve informace o dvou nejúspěšnějších akciích a poté o dvou nejneúspěšnějších akciích. Pro klesající burzu naopak. Podmínkou „PX_CHANGE > 0“ tedy určíme, zda burza klesala, či stoupala a nadefinujeme list (jelikož se na tyto bloky budeme odkazovat, pojmenujeme si list jako „TOP 2 – nejlepší“) obsahující text uvozující název první a druhé nejlepší akcie (oddělené spojkou „a“) a ukončené tečkou:

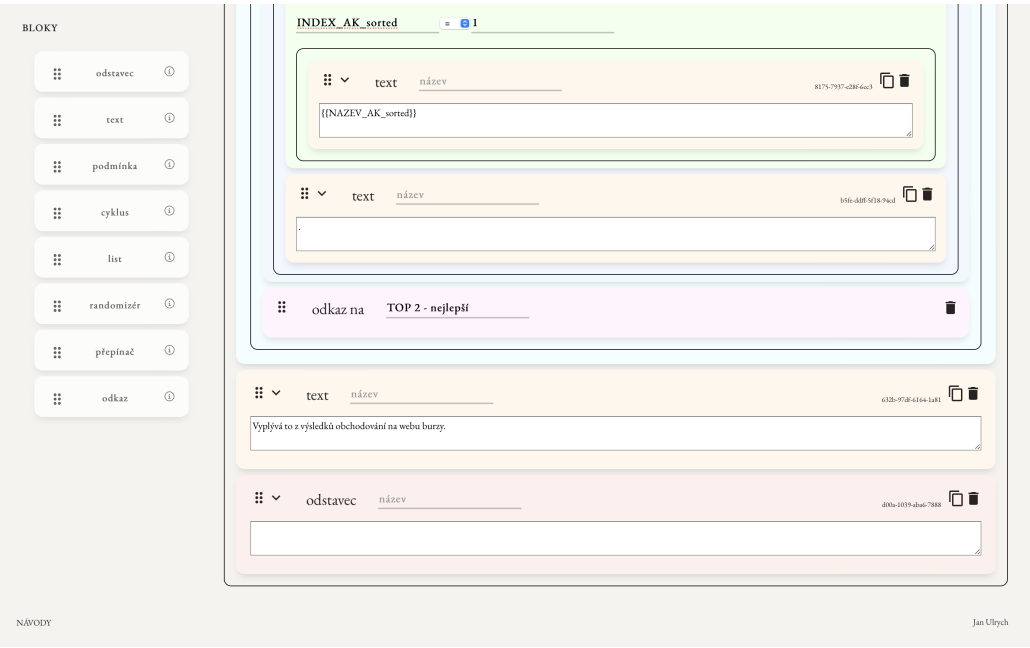




B.2.3.5 5. krok:

Za list vložíme odkaz na list „TOP 2 – nejhorší“. Ten nyní implementujeme analogicky jako list dvou nejlepších akcií, jen pro opačné hodnoty. Na konec ještě vypíšeme text a ukončíme odstavec:

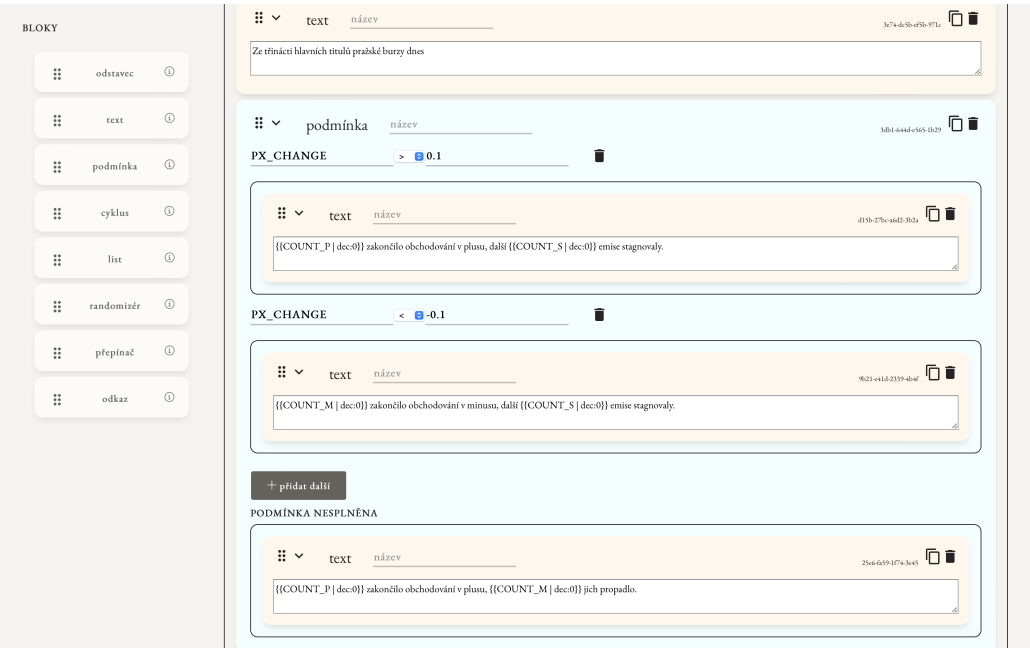




B.2.3.6 6. krok:

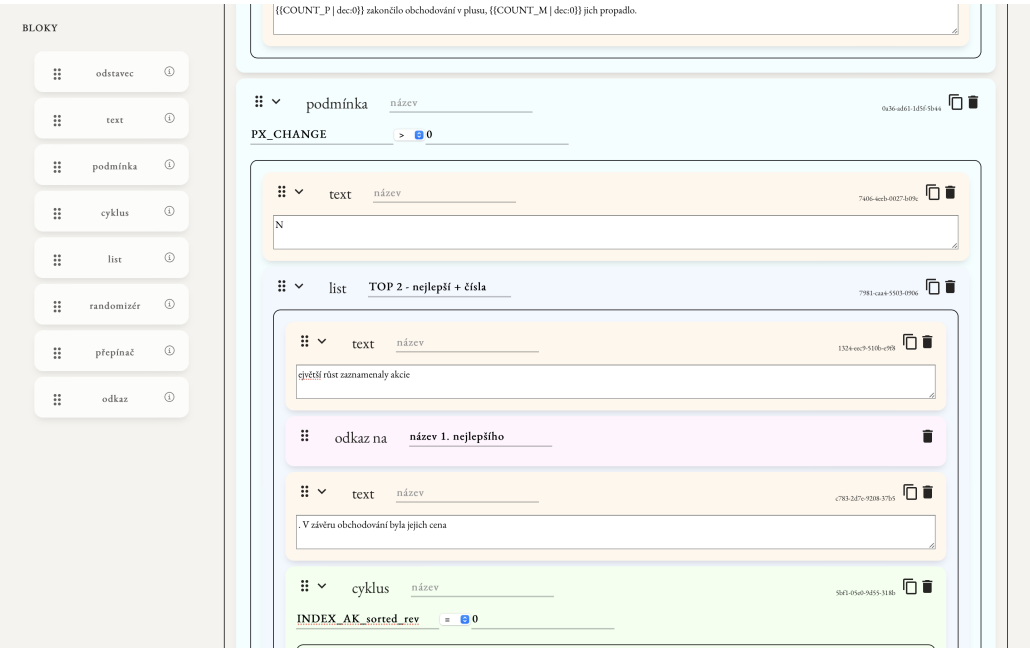
Následující odstavec bude obsahovat podobný text, rozšířený ale o tvrdá data a další informace. Na začátku tohoto odstavce vypíšeme informace o počtu rostoucích, klesajících a stagnujících akcií. K tomu si nadefinujeme akci pro předzpracování dat „COUNT“ s výsledkem „COUNT_P“, která pro sloupec „KURZ_AK“ vypočítá

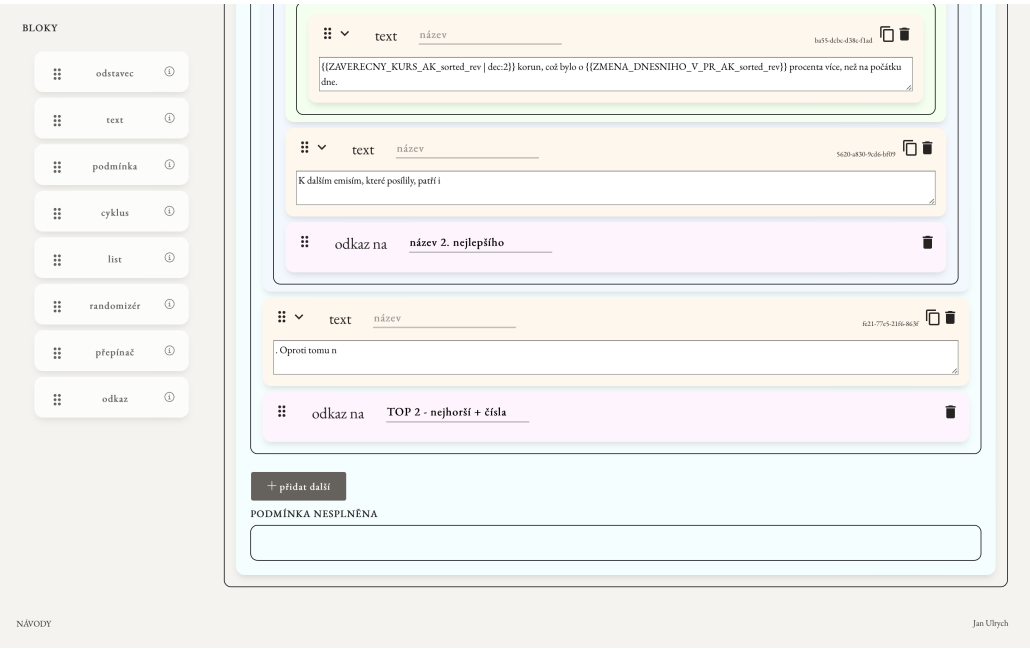
výskyt hodnot „P“ (akcie v plusu). Obdobně pro hodnoty „S“ (akcie stagnuje) a „M“ (akcie v minusu). Poté už pouze na základě pohybu burzy vypíšeme příslušný text:



B.2.3.7 7. krok:

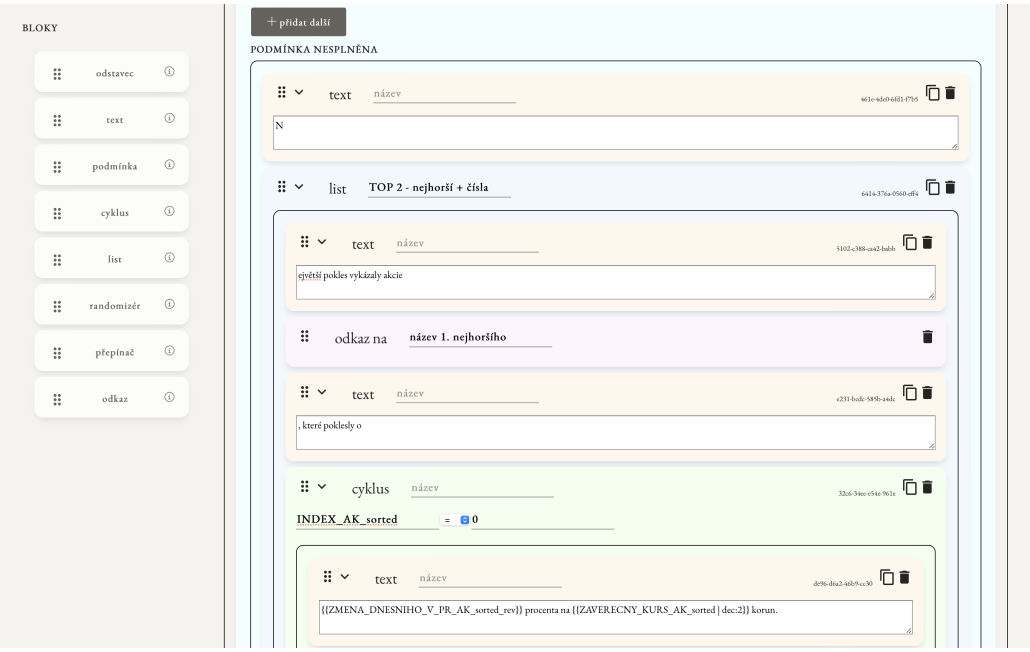
Následuje podobná informace, jako v předchozím odstavci, ale rozšířená o podrobnější data. Pro první nejlepší akcii vypíšeme i její posun oproti začátku obchodního dne. Generování bude opět rozděleno podle aktuálního pohybu burzy, bloky vložíme do listu, na který se poté budeme odkazovat z větve „podmínka nesplněna“ při výpisu akcií v opačném pořadí.

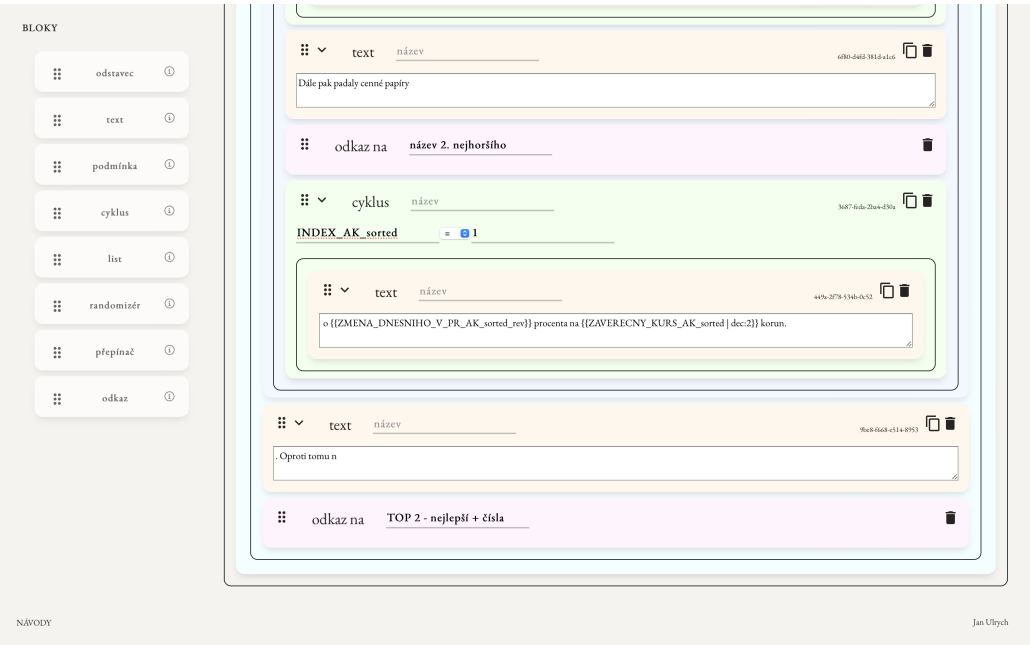




B.2.3.8 8. krok:

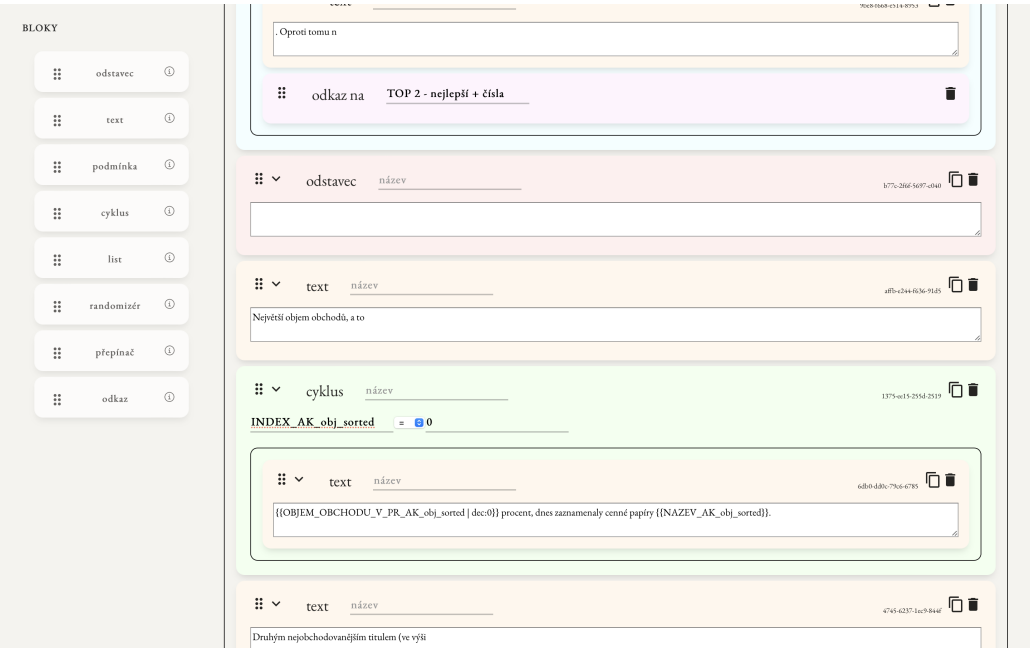
Obdobně jako pro stoupající burzu, implementujeme informace i pro klesající burzu, tedy pro větev podmínka nesplněna:





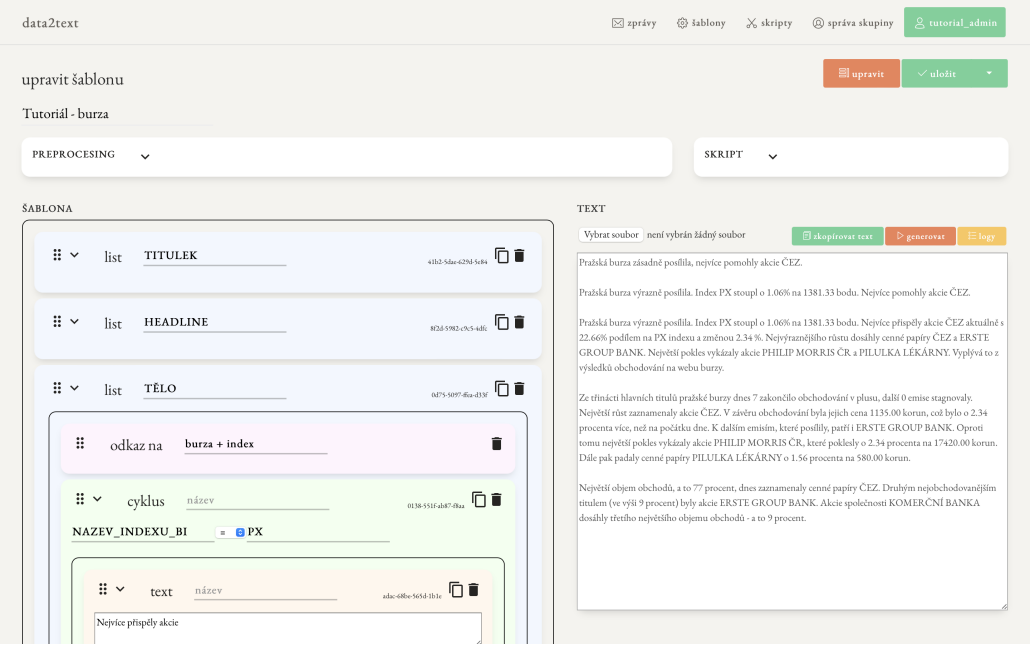
B.2.3.9 9. krok:

V posledním odstavci budeme vypisovat informace o nejobchodovanějších akciích na burze. K tomu využijeme data ze souboru „output_ak_obj_sorted.csv“. Konkrétně v cyklech budeme hledat pro sloupec „INDEX_AK_obj_sorted“ hledat hodnoty 0, 1 a 2. Pro ně vypíšeme příslušný text:



B.2.3.10 10. krok:

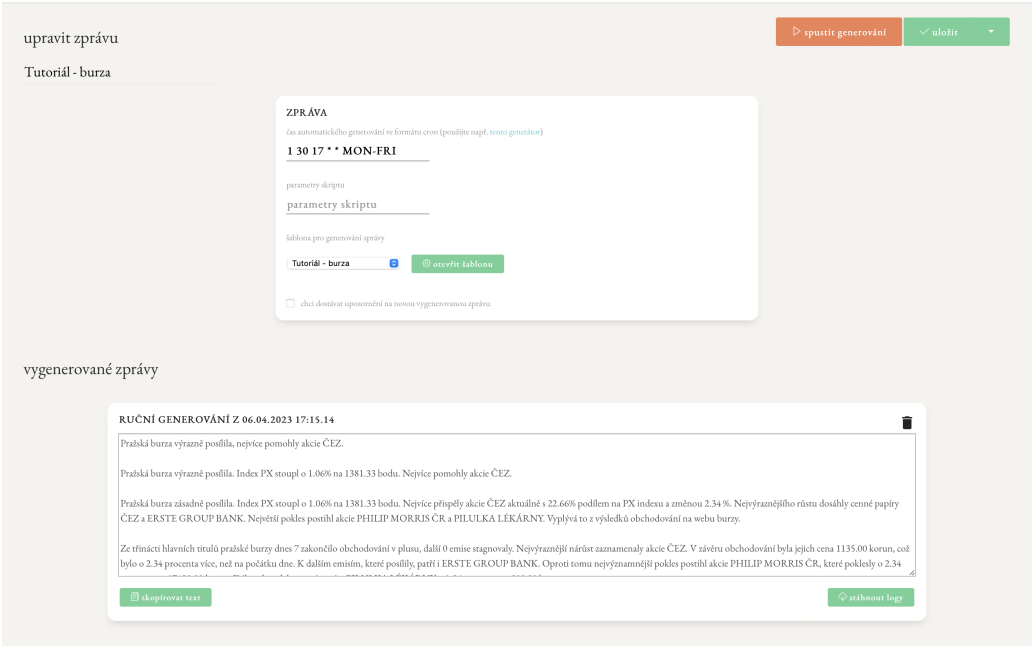
Tím máme hotovou šablonu pro tělo. Pro větší přehlednost opět nové bloky vložíme do listu „TĚLO“ a vygenerujeme si text:



B.3 Zpráva

Pro automatické generování zprávy v daný čas a uchování výsledků generování musíme definovat zprávu, které předáme čas, ve který se má spustit generování textu a šablonu, podle které chceme text generovat.

Novou zprávu si pojmenujeme například jako “Tutoriál – burza“. Čas automatického generování nastavíme na hodnotu „1 30 17 * * MON-FRI“ (tento řetězec nám říká: text se bude generovat v pondělí až pátek, každý měsíc, každý den v 17 hodin, 30 minut a 1 vteřinu). Parametry skriptu necháme prázdné. Jako šablonu vybereme námi dříve vytvořenou šablonu – „Tutoriál – burza“. Pokud budeme chtít dostávat upozornění na nově vygenerovaný text, zaškrtneme pole pod výběrem šablony. Můžeme vyzkoušet zprávu vygenerovat ručně kliknutím na tlačítko „spustit generování“.


upravit zprávu
spustit generování
uložit
Tutoriál - burza
ZPRÁVA
čas automatického generování ve formátu cron (použijte např. tento generátor)
1 30 17 * * MON-FRI
parametry skriptu
parametry skriptu
šablona pro generování zprávy
Tutoriál - burza
otevřít šablonu
chci dostávat upozornění na novou vygenerovanou zprávu
vygenerované zprávy
RUČNÍ GENEROVÁNÍ Z 06.04.2023 17:15.14
Pražská burza výrazně posílila, nejvíce pomohly akcie ČEZ.
Pražská burza výrazně posílila. Index PX stoupl o 1.06% na 1381.33 bodu. Nejvíce pomohly akcie ČEZ.
Pražská burza zásadně posílila. Index PX stoupl o 1.06% na 1381.33 bodu. Nejvíce přispěly akcie ČEZ aktuálně s 22.66% podílem na PX indexu a změnou 2.34 %. Nejvýraznějšího růstu dosáhly cenné papíry ČEZ a ERSTE GROUP BANK. Největší pokles postihl akcie PHILIP MORRIS ČR a PILULKA LÉKÁRNY. Vyplývá to z výsledků obchodování na webu burzy.
Ze třinácti hlavních titulů pražské burzy dnes 7 zakončilo obchodování v plusu, další 0 emise stagnovaly. Nejvýraznější nárůst zaznamenaly akcie ČEZ. V závěru obchodování byla jejich cena 1135.00 korun, což bylo o 2.34 procenta více, než na počátku dne. K dalším emisím, které posílily, patří i ERSTE GROUP BANK. Oproti tomu nejvýznamnější pokles postihl akcie PHILIP MORRIS ČR, které poklesly o 2.34
zkopírovat text
stáhnout logy