Západočeská univerzita v Plzni
Fakulta Aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Webová aplikace pro vizualizaci časové osy

Plzeň, 2017 Daniel Holubář

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 28. června 2017

Daniel Holubář

Abstract

The topic of bachelor thesis is demonstration of timeline created on Department of Computer Science and Engineering of Faculty of Applied Sciences of the University of West Bohemia in Pilsen. The theoretical part deals with related work created in previous years and with the design of new web application which demonstrates the use of the timeline for educational purposes. The practical part describes implementation of this application. It allows creation of new data for timelines, creation of new tests and completition of these tests.

Abstrakt

Předmětem bakalářské práce je demonstrace časové osy vytvořené na Katedře informatiky a výpočetní techniky Fakulty Aplikovaných věd Západočeské univerzity v Plzni. Teoretická část se zabývá souvisejícími pracemi vytvořenými v předchozích letech a návrhem nové webové aplikace, která předvádí využití časové osy pro výukové účely. Praktická část popisuje implementaci aplikace, která umožňuje vytváření nových dat pro časové osy, nových testů a jejich vyplňování.

Poděkování

Tímto bych rád poděkoval Ing. Richardu Lipkovi, Ph.D. za trpělivost, ochotu a užitečné rady, díky kterým jsem zvládl tuto bakalářskou práci vypracovat.

Obsah

1 Úvod

Již od pravěku lidé zaznamenávali různé události. Od maleb v jeskyních přes kamenné desky, hliněné tabulky, papyrové svitky, ručně psané a později tištěné knihy až po dnešní dobu elektronických informačních zdrojů. Internet se stal studnicí informací pro většinu lidstva a představuje nejpřístupnější zdroj téměř jakéhokoliv vědění. Mezi nejdůležitější části tohoto vědění patří historie lidstva. Umožňuje nám získat povědomí o předcích, poučit se z chyb předchozích generací nebo čerpat vědění od největších mozků naší civilizace.

Historie lidstva je velmi pestrá a komplexní. Při množství údajů je důležité neztrácet přehled a souvislosti mezi různými událostmi. Tato práce je součástí většího projektu vytvořeného na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské Univerzity, jehož cílem je vytvoření časové osy, která zachycuje historické údaje spolu se souvislostmi mezi nimi přehlednou a jasně srozumitelnou formou.

Práce staví na hotových pracích studentů Západočeské Univerzity. Slečna Gabriela Hessová zpracovala problematiku automatického získávání historických údajů z webových zdrojů [9]. Pan David Hrbáček se zabýval zpracováním těchto údajů a ohodnocením jejich důležitosti při zobrazení uživateli [10]. Pan Michal Kacerovský [11] potom vytvořil nástroj pro přehledné zobrazení všech získaných dat a souvislostí mezi nimi pomocí časové osy. Pan David Merunko potom zpracoval problematiku generování, udržování a zpracování časové osy pomocí samostatné aplikace v programovacím jazyku Java [12].

Cílem práce je vytvoření aplikace, která naváže na předchozí práce. Aplikace využije hotového widgetu pana Kacerovského a částečně rozšíří jeho funkcionalitu. Dále widget zasadí do aplikace, která umožní jeho použití pro vytváření nových dat pro časové osy, vytváření testů a spouštění těchto testů. Aplikace bude sbírat statistické údaje, pomocí kterých půjde analyzovat, zda je tento projekt prospěšný pro výuku historie.

2 Technologie pro tvorbu webových aplikací

Díky rychlému rozvoji existuje mnoho technologií, které umožňují tvorbu webových aplikací. Mezi těmito je vhodné vybrat takové technologie, díky kterým aplikace bude rychlá, kvalitní a robustní. Pro bakalářskou práci jsem kromě základních technologií pro tvorbu webů (HTML, CSS, JavaScript) vybral následující technologie:

- **MySQL** - SQL - Structured Query Language
- **Laravel** - Framework nad jazykem PHP

2.1 MySQL

MySQL je databázový systém postavený na jazyku SQL, který uplatňuje relační databázový model. Mezi přednosti této technologie patří zejména rozšiřitelnost, uživatelská podpora a robustnost. Jedná se o databázový systém, který zná naprostá většina programátorů. Komunikace s touto databází probíhá pomocí SQL dotazů. Z porovnávání MySQL a PostgreSQL [6] vyplývá, že při určitých operacích má PostgreSQL rychlejší dobu zpracování než MySQL, nicméně MySQL lépe zvládá práci s velký množstvím dat a obsluhováním více procesů najednou. Na tom má zásluhu zejména zvýšení počtu podporovaných jader pro MySQL databázi z 4 na 64.

2.2 PHP

Jedná se o velmi rozšířený interpretovaný skriptovací jazyk, který se používá k tvorbě dynamických webových aplikací. Dynamické aplikace jsou zpracovány serverem. To znamená, že PHP skript je spuštěn na straně serveru a jeho výsledek je poté poslán uživateli. Je to velmi rozšířený jazyk, jehož syntax je inspirována několika staršími programovacími jazyky, například C a Perl. Obsahuje velké množství knihoven pro práci s různými databázovými systémy, dále knihovny pro zpracování grafiky, přenos a ukládání souborů a zpracování textů. I když podle tiobe indexu [8] je PHP k dnešnímu dni na jednom z posledních míst mezi jazyky používanými pro tvorbu webových aplikací, stále se jedná o nejpodporovanější jazyk mezi hostingy, který má velkou komunitu a díky tomu i uživatelskou podporu. S novou verzí PHP (PHP 7), která přinesla nové funkce a optimalizace, se obliba tohoto jazyka pomalu vrací.

2.2.1 PHP Framework - Laravel [13]

Tento framework patří mezi nejoblíbenější nástroje vybírané pro tvorbu PHP aplikací. V posledních letech zažívá rozmach proti ostatním známým frameworkům [5]. Mezi jeho výhody patří zejména rychlé vytvoření základní aplikace, která v sobě již obsahuje základní správu uživatelů. Pomocí širokého spektra příkazů pro příkazovou řádku [3] lze rychle vytvořit potřebné modely, controllery, migrace pro databázi

a také vložení základních dat do databáze. Programátor se tak nemusí zabývat ruční tvorbou tříd nebo tabulek v databázi. Vše potřebné se tvoří automaticky.

3 Stávající nástroje pro vizualizaci časové osy

Existující řešení zkoumal ve své práci již Michal Kacerovský. Bude tedy vhodné rozdělit seznam stávajících řešení na ta již dříve zkoumaná a ta, které existují v dnešní době.

3.1 Řešení zkoumaná Michalem Kacerovským

3.1.1 TimelineJS

Jak bylo uvedeno výše, jedná se o službu, která podle vstupních dat vygeneruje časovou osu. Během vytváření časové osy mu byla tato knihovna inspirací. Tato služba je stále aktivní.

3.1.2 Dipity

Služba Dipity byla obdobná jako služba TimelineJS s tím rozdílem, že uživatelé museli být registrováni. Tato služba již není aktivní.

3.1.3 Timglider

Oproti předchozím produktům nabízí Timeglider možnost využití jQuery widgetu se základními nástroji pro práci s osou, jako je vkládání nových záznamů, editace nebo přiblížení časové osy pomocí kolečka myši. Timeglider je stále aktivní.

3.1.4 vis.js

Vis.js je knihovna s mnoha různými službami, mezi nimiž je i služba vizualizace časovou osou. Vis.js nabízí i možnost forku, díky čemuž si ji může uživatel přizpůsobit pro své účely, nebo rozšířit podle svých představ. Tato knihovna je stále aktivní.

3.1.5 Timeline (SimileWidgets)

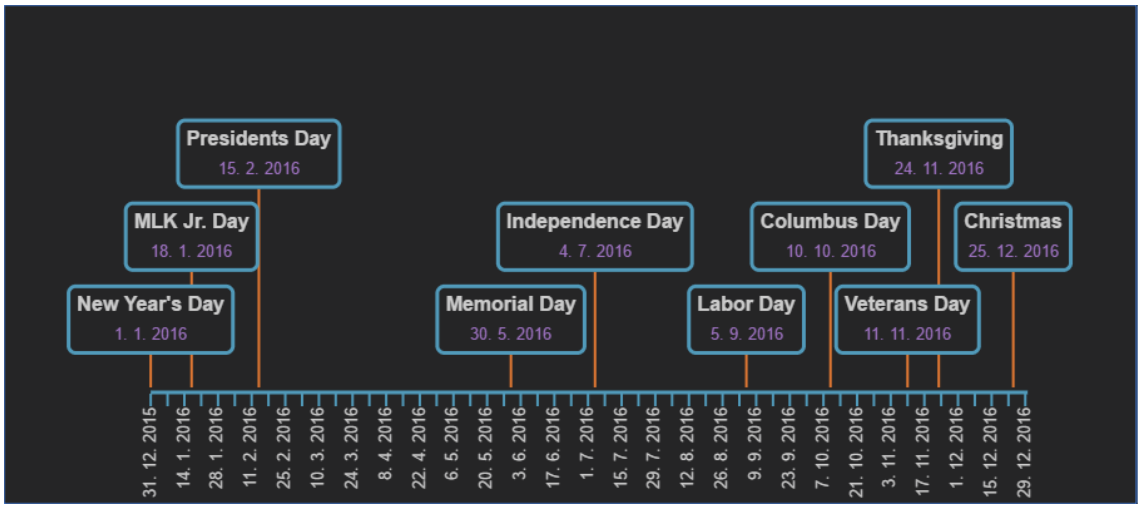
Vývoj této knihovny byl ukončen již v roce 2009. Tento stav stále trvá.

3.2 Řešení dostupná v dnešní době

Díky rozvoji moderních internetových technologií jako jsou HTML5 a CSS3 je množství služeb a knihoven pro zobrazení časové osy obrovské. Z toho důvodu uvedu jen několik pro mne nejzajímavějších.

3.2.1 GoJS

Knihovna GoJS nabízí velké množství nástrojů a možností vizualizace. Jednou z nich je i časová osa. Obsahuje mnoho návodů a ukázek, díky čemuž je její použití snadné. Pro osobní účely je tato knihovna poskytována zdarma, pro komerční využití je třeba zaplatit licenční poplatek.



Obr. 1: Příklad použití knihovny GoJS

Knihovna disponuje přehlednou dokumentací a jednoduchou implementací. Data se vkládají pomocí JSON formátu a osa samotná se dá velmi jednoduše roztahovat nebo naopak smršťovat. Záznamy na ní včetně časových údajů se dynamicky přizpůsobují těmto změnám. Po ose se dá pohybovat myší nebo klávesami.

Do časové osy není možné vkládat obrázky a osa nelze přibližovat ani oddalovat. Jednotlivé záznamy není možné žádným způsobem spojovat.

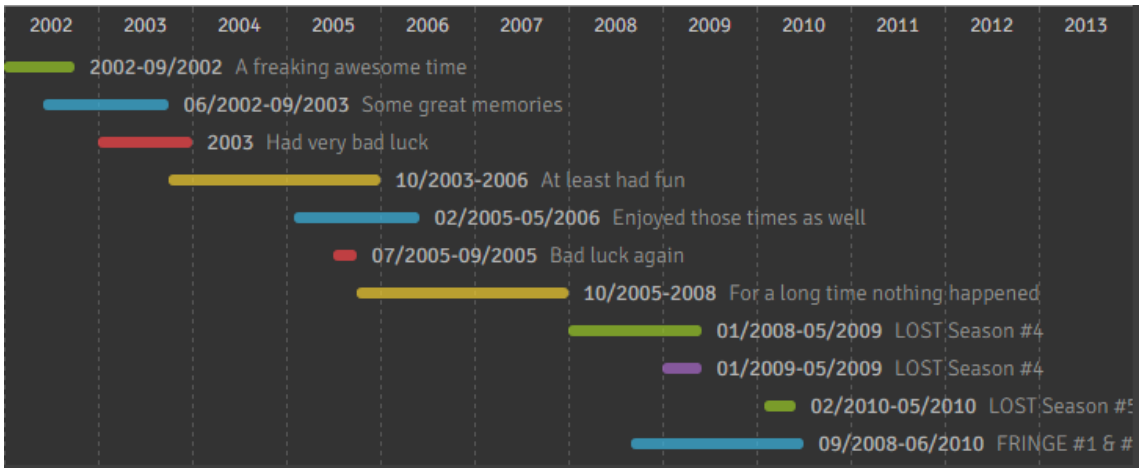
Knihovna GoJS je užitečným nástrojem pro jednoduché a přehledné časové osy, avšak díky absenci některých nástrojů a možností se stává nevhodnou pro aplikaci na časové ose zpracovávané panem Kacerovským.

Vydavatel	Northwoods Software Corporation
URL	http://gojs.net
Licence	https://gojs.net/latest/doc/license.html
Aktuální verze	1.7

Tab 1: Informace o knihovně GoJS

3.2.2 Timesheet.js

Knihovna Timesheet.js nabízí grafickou vizualizaci časové osy pomocí HTML5 a CSS3. Optimalizace na mobilní zařízení je pro tuto knihovnu samozřejmostí.



Obr. 2: Příklad použití knihovny Timesheet.js

Příklad použití knihovny je znázorněn na Obr. 2. Knihovna umožnuje jen znázornění časových úseků, a i když přehledně a se zajištěním nepřekrývání jednotlivých událostí. Knihovna bohužel nepodporuje další funkce jako vloažení obrázku, spojování událostí ani manipulaci s časovou osou (zoom) a to ani pomocí kláves.

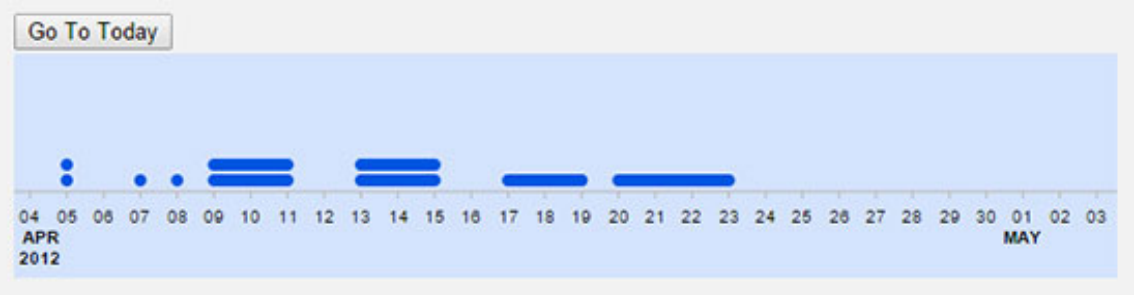
I přes moderní zpracování a přehledný design je tato knihovna pro naše účely kvůli výše uvedeným nedostatkům nepoužitelná.

Vydavatel	Sebastian Müller
URL	https://sbstjn.com/timesheet.js
Licence	MIT
Aktuální verze	—

Tab 2: Informace o knihovně Timesheet.js

3.2.3 Chronoline.js

Chronoline.js, stejně jako přechozí knihovna, umožnuje zobrazení časové osy pouze v jednoduché formě, přestože oproti předchozí nabízí více možností a nástrojů.



Obr. 3: Příklad použití knihovny Chronoline

Knihovna nabízí ovládání pomocí myši, dále umí i zoomovat, k čemuž však potřebuje externí formulář, do kterého se vloží číselná hodnota zoomu. Tato knihovna také umí vkládat dva druhy událostí. Jedny se zobrazují jako barevné pásy o výšce celé časové osy. Jedná se typicky od dlouhodobé události. Kratší události jsou zobrazovány pomocí čar se zaoblenými konci. Podle informací na stránce se jedná primárně o nástroj vytvořený na míru pro stránky https://zanbato.com/.

Této knihovně opět scházejí nástroje, které bychom potřebovali pro naši časovou osu, tedy spojování událostí, ovládání pomocí klávesnice nebo zobrazení dodatečných informací při najetí na událost.

Vydavatel	StoicLoofah
URL	http://stoicloofah.github.io/chronoline.js/
Licence	MIT
Aktuální verze	—

Tab 3: Informace o knihovně Chronoline.js

I přes množství knihoven a služeb pro tvorbu časových os se mezi nimi nevyskytuje žádná, která by splňovala dané požadavky - přehledné zobrazení dat včetně souvislostí mezi těmito daty. Výše zmíněné příklady byly vybrány s cílem poukázat právě na tyto nedostatky i přes použití moderních technologií a nejnovějších postupů.

4 Souhrn ostatních prací

V této sekci se budu zabývat tím, co bylo doposud v rámci projektu vytvořeno ostatními studenty, tedy pracemi slečny Gabriely Hessové a pánů Davida Hrbáčka, Michala Kacerovského a Davida Merunka.

4.1 Automatické získání historických údajů z webu [9]

Cílem práce Gabriely Hessové bylo získání dat z internetových zdrojů a jejich zpracování pro aplikace, které s těmito daty pracují. Práce se tedy zabývá metodami získávání dat z textu z různých webových zdrojů. Jako hlavní zdroj pro data byla vybrána Wikipedie.

V druhé části práce se autorka zabývá praktickou implementací nástroje pro získání dat z webových zdrojů. Soubory s daty s Wikipedie mají velikost v řádech GiB(46,7 GiB v době tvroby práce slečny Hessové [9, str. 11]). To má za následek prodlevy při vyhledávání v datech, které autorka řeší pomocí indexu. Jedná se o pomocnou datovou strukturu, díky které je vyhledávání rychlejší [9, str. 21].

Pro získání dat není využit žádný již realizovaný nástroj jako je SAX nebo DOM, autorka problém řeší vlastním nástrojem pro čtení dat a jejich zpracování. Při zpracování se data ukládají do relační databáze, v dalším kroku se data převedou do grafové databáze vytvořené panem Merunkem.

4.2 Generování a vizualizace časové osy [12]

Cílem práce Davida Merunka bylo vytvoření grafové databáze, která by byla uložena na serveru. Dále se zabývá vytvořením rozhraní pro přístup k této databázi a nakonec vytvorenim aplikace pro zobrazení časové osy pomocí dat uložených na serveru.

Při práci s grafy je nutné správně volit algoritmy, zejména pro určení metrik jako jsou přehled o náročnosti vyhledávání a informace o velikosti grafu. Autor v práci metriku využívá pro určení relevantnosti vrcholů vůči sobě [12, str. 23]. Pro určení důležitosti vrcholu potom využívá stupeň vrcholu, tedy počet hran, které vedou do/z vrcholu [12, str. 23]. Autor dále uvádí principy a mechaniky grafové databáze, její rozdíly proti relačním databázím a porovnává je z hlediska výkonnosti [12, str. 26].

Jako databáze byla vybrána Neo4j. Jedná se o grafovou databázi, která má všechny potřebné vlastnosti. Při procházení databáze se dá vybrat ze dvou přístupů: do hloubky a do šířky.

Pro databázi je nutné také správně určit typy vrcholů, čemuž se autor věnuje v kapitole 7.2 [12, str. 34]. Vrcholy jsou vybrány čtyři.

- **Person** - typ vrcholu pro důležitou historickou osobnost, který udržuje vlastnosti spjaté s životem osobnosti [12, str. 35].
- **Event** - typ vrcholu pro důležité historické události a data s nimi spjatá. [12, str. 35]
- **Place** - typ vrcholu pro důležitá historická místa a politické útvary. Tento typ vrcholu není zobrazen na časové ose, nicméně slouží jako obalovací vrstva pro ostatní vrcholy [12, str. 35].
- **Item** - typ vrcholu pro historické vynálezy, dokumenty a technologie [12, str. 34].

K těmto vrcholům jsou dále navrženy vztahy, aby co nejlépe definovaly souvislosti mezi jednotlivými vrcholy.

- **Relationship** - vztah pro vrchol typu Person nebo Place, tedy například pro definování vztahu mezi matkou a potomkem, nebo přerod státu do jiného státu [12, str. 37].
- **Interaction** - tento typ vztahu byl vytvořen pro definování souvislostí mezi dvěma vědci, kteří spolu spolupracovali, nebo státníky, kteří jednali o důležitých dohodách mezi svými státy [12, str. 37].
- **Participation** - tento vztah představuje účast osoby na události nebo na určitém místě [12, str. 37].
- **Creation** - definuje vznik místa, vynálezu nebo narození důležité osoby nebo počátek události [12, str. 37].
- **Cause** - vztah definuje příčinu vzniku vrcholu ve vztahu k jinému vrcholu [12, str. 37].
- **Part_of** - tento typ vztahu definuje vrcholy, které jsou součástí většího celku [12, str. 37].
- **Takes_place** - slouží pro určení vztahu mezi událostí a místem, kde se událost konala [12, str. 37].

Databázové API je zodpovědné za ukládání, načítání a vyhledávání dat. Autor bere v úvahu i možnosti vyhledávání. Za tímto účelem byly vytvořeny odpovídající metody, které jsou blíže popsány v [12, str. 40]. Tyto metody zajišťují vytvoření databáze, načtení databáze a odpojení databáze. Obsahují logiku pro prohledávání grafů podle zadaných parametrů, vkládání vrcholu, editaci vrcholu, odstranění vrcholu. Metody pro vkládání, editaci a odstranění jsou vytvořené i pro hrany. API obsahuje i další pomocné metody pro manipulaci s grafem nebo hranou.

4.3 Zpracovnání časových údajů pro jejich vizualizaci [10]

Cílem práce Davida Hrbáčka bylo vytvoření algoritmu pro ohodnocování částí grafu podle hodnot zadaných uživatelem. Dále měl autor vytvořit REST server, který by komunikoval s grafovou databází, webovým klientem a javovskou aplikací.

Jako jedna z možných metrik pro odhodocování částí grafu se může používat tzv. centralita uzlu. Tato metrika se rozděluje na následující druhy:

- **Degree centrality** - u této metriky je důležitost uzlu určena jeho stupněm, tedy čím více sousedů vrchol má, tím důležitější je [10, str. 5].
- **Closeness centrality** - tato centralita je založena na vzdálenosti vrcholu od ostatních uzlů, kdy s větší vzdáleností klesá jeho důležitost [10, str. 5].
- **Betweenness centrality** - tato metrika je založena na počtu nejkratších cest mezi dvěma vrcholy, na kterých leží pozorovaný vrchol [10, str. 5].

Úpravou v používání výše zmíněné metriky získáme ohodnocení grafu pomocí sady kritérií [10, str. 5]. Tato metoda určuje důležitost a celkovou úlohu uzlu v rámci grafu podle jeho vlastností.

Poslední možností ohodnocení uzlu v práci je PageRank [10, str. 5]. Právě tento algoritmus si autor vybral k implementaci pro účely své práce. Algoritmus patří mezi známější zejména díky společnosti Google, která ho využívala pro ohodnocování důležitosti webových stránek. Výhodou tohoto algoritmu oproti předchozím metrikám je, že nebere vrchol grafu samostatně, ale v souvislosti s okolními vrcholy.

Knihovna pro práci s grafem je inspirována již existující knihovnou GraphStream. Knihovna je tedy rozdělena do dvou samostatných celků - graph a graph-ranking. První slouží k implementaci grafu a druhý poté obsluhuje ohodnocování [10, str. 23].

Knihovna graph je koncipována tak, že hrana i uzly jsou reprezentovány rozhraním, což zamezuje možnost zdvojení hran nebo vrcholů. V knihovně je též ošetřena potřeba otestování, zda hrana propojuje existující vrcholy. Samozřejmostí jsou potom metody pro vkládání a mazání vrcholů a hran. Graf je tedy reprezentován pomocí rozhraní, což reflektují třídy a rozhraní implementující knihovnu [10, str. 26].

Pro samotný algoritmus PageRank byla vybrána možnost výpočtu pomocí spojových seznamů. Na rozdíl od maticového výpočtu se při tomto výpočtu počítá důležitost každého uzlu zvlášť, což je výhodné zejména kvůli řídkosti matice přechodů používané ve výpočtu. Algoritmus taktéž zahrnuje i uživatelem zadané priority.

Knihovna pro vnější přístup obstarává komunikaci s databází a s klientem. Uživatel

má možnost zadat parametry pro zobrazení grafu historických událostí. Knihovna data od uživatele vezme a zavolá metodu, jejíž parametry jsou právě data od uživatele. Metoda potom vrací graf. Mezi další možnosti patří získání podgrafu, vložení hrany či vrcholu a získání dat uložených ve vrcholu.

Vytvořený REST server poskytuje data ve formátu JSON, což má kladný vliv na rychlost komunikace. Metody serveru kopírují metody knihovny pro vnější přístup. Server tedy získá data z knihovny, převede je do formátu JSON a následně odešle.

Samotná implementace serveru byla realizována pomocí frameworku Spring MVC 4. Důvodem výběru bylo:

- Snadná implementace REST rozhraní pomocí anotací.
- Hotové řešení pro JSON formát.
- Získávání objektů pomocí návrhového vzoru IoC.
- Velká základna uživatelů.

4.4 Vizuální reprezentace precedenčního grafu [11]

Cílem práce Michala Kacerovského je vizuální reprezentace časové osy pomocí webového widgetu. Práce navazuje na práce zmíněné výše. Jedná se o koncový produkt celého zpracování historických dat.

Nejprve je nutné definovat časovou osu a upřesnit její vlastnosti spolu s požadavky, které jsou na ní kladené [11, str. 3]. Protože práce úzce navazuje na práci pana Merunka, Využívá pan Kacerovský stejný typ vrcholů a vztahů které, jsou definovány v analýze práce pana Merunka.

Jak autor popisuje v kapitole 4.1 [11, str. 8], vývoj webového widgetu grafického charakteru může být chápán jako analogie k grafickým aplikacím realizovaným pomocí jiných programovacích jazyků, například Java a C++. Tím je myšleno, že tak jako grafické aplikace využívají rozdělení na vrstvy, kdy jedna vrstva mapuje data na objekty, další vrstva obstarává grafickou podobu a poslední vrstva rozhoduje o vykreslování, tedy co má uživatel vidět, tak stejným způsobem můžeme realizovat i webový widget grafického charakteru. Pomocí canvasu může programátor prezentovat jakoukoliv grafiku, ovšem za cenu ztráty možnosti interakce, protože vyobrazení převádí data na pixely. Naproti tomu je zde možnost použití technologie SVG, která vytvoří mimo jiné vrstvu mezi programátorem a vykreslováním pixelů. Protože tato technologie pracuje s DOM, je zde pro programátora stále možnost zachycení akcí nad vykreslovanými objekty [11, str. 8].

Při práci s časovou osou je nutné dát si pozor na určité věci spojené s časem [11, str. 9]. Například ISO 8601, což je norma pro časový formát, nebere rok 0 v potaz, to znamená, že tento rok neexistuje v pojetí zobrazení času. Mezi další problémy patří například letní a zimní čas a další podobné anomálie, o kterých se autor podrobně zmiňuje ve své práci.

Pro tvorbu své práce autor vybírá HTML5 a CSS3. Jedná se o nejnovější verze těchto technologií, které disponují mnohými nástroji, jenž předčí první dvě zmiňované technologie. Podstatnou podmínkou je v neposlední řadě také to, aby byl widget responzivní, tedy aby dokázal reflektovat hustotu pixelů a rozlišení zobrazovacího zařízení, a dokázal tak data zobrazovat stále přehledně. Protože widget komunikuje se serverovou částí, musí být stanoven způsob přenosu dat. V dřívější práci byl vybrán formát JSON, čímž je formát fakticky stanoven a autor to musí vzít v potaz.

Aby bylo grafické rozhraní přehledné a dobře se v něm orientovalo, je jako způsob grafického zobrazení vybrán systém se souběžnými pásy, kdy každý pás reprezentuje jiný typ události. Události jsou dále chronologicky seřazené, což je předpoklad samotné časové osy. Data jsou řazena od nejstarších po nejnovější ve směru zleva doprava. Mezi událostmi, které jsou vůči sobě nějakým způsobem závislé, je jako reprezentace tohoto vztahu zobrazena čára. Protože kompletní zobrazení vztahů by bylo nepřehledné, má uživatel možnost procházet podmnožinu těchto vztahů [11, str. 14]. Jako ovládací prvky jsou vybrány logicky klávesnice a myš. Uživatel má možnost osu ovládat pomocí myši kolečkem pro přiblížení a metodou drag and drop může osu posouvat podle své potřeby. Na klávesnici slouží pro přiblížení klávesy + a -, zatímco pro posun osy se využívají tlačítka kurzorových šipek.

Nedílnou součástí widgetu je možnost filtrování informací. Filtry nejsou součástí vykreslovacího widgetu, a musejí být realizovány jako prostředek, který přistupuje přímo k datům, která widget zobrazuje. Autor dále uvádí některé možnosti filtrování [11, str. 16].

V další části autor navazuje na práce kolegů a stejně jako oni i on provádí výčet existujících služeb pro vykreslování časové osy a diskutuje jejich přednosti a nedostatky. Stejně jako oni i on dochází k závěru, že pro svou práci navrhne knihovnu novou tak, aby vyhovovala všem stanoveným požadavkům. Dále widget zapojuje mezi práce ostatních, protože David Hrbáček [10] a David Merunko [12] vytvořili další části z celého projektu, konkrétně určování priorit a důležitosti vrcholů a hran a databázi pro ukládání grafu, který reprezentuje tuto časovou osu.

V kapitole uživatelského rozhraní [11, str. 25] se autor zabývá rozložením a vzhledem samotného widgetu. Jak již bylo zmíněno, aplikace bude zobrazovat více souběžných pásů pro různé druhy událostí. Je nutné, aby pásy reflektovaly i souběžné události a přizpůsobovaly se svému obsahu. Samotná časová osa pro přehled časového charakteru je umístěna ve spodní části widgetu. Je nutné, aby začátek a konec zobrazo-

vaných dat správně začínal a končil i podle časové osy. Nástroj je tedy vytvořen tak, aby zohledňoval různá rozlišení a přibližování a oddalování. Autor detailně popisuje způsob řešení tohoto problému [11, str. 25] včetně výpočtu zobrazení. Plocha celého widgetu je zhruba trojnásobná ploše viditelné na zobrazovacím zařízení. Autor tedy řeší i to, že některé věci vidět nejsou, a není tedy třeba je zobrazovat uživateli [11, str. 32]. Případné souběžné události jsou řešeny pomocí Lane algoritmu. Ten obstarává to, aby se souběžné události nepřekrývaly a aby události byly správně řazeny [11, str. 33]. Aby se pásy správně vykreslovaly, jsou obaleny do skupiny pásů. Při posouvání a přibližování osy se výpočty pásů provádějí přes tuto skupinu. Jejím dalším úkolem je i distribuce volného místa. To znamená, že se zjistí, kolik prostoru pro vykreslení je na šířku a výšku, a tento prostor je rozdělen mezi všechny pásy. Jako abstrakce události je uvedena položka pásu. Tato položka je svázána s objektem, který nese data pro tuto položku na časové ose. Každá událost je poté rozdílným způsobem vykreslena podle svých vlastností a svého typu, aby měl uživatel na první pohled přehled [11, str. 35]. Součástí vykreslení jsou i vztahy. Ty, jak již bylo řečeno dříve, nejsou vykreslovány najednou, ale vykreslí se vždy podmnožina podle výběru událostí uživatelem. Součástí zpracování je i různé pojetí zoomu podle použití klávesnice nebo myši. Při použití klávesnice se mění měřítko celého zobrazení. Při použití myši se potom bere v úvahu i její pozice, kdy pozice kurzoru určuje střed celé transformace. Předpokládá se, že když je myš na nějaké události, má při přiblížení nebo oddálení být tato pozice brána jako střed. Další funkcí widgetu je zaostření. To funguje tak, že pokud je nějaká událost vybrána pro zaostření, widget postupně zoomuje do té doby, než využije maximální prostor pro zobrazení dané události.

Důležitou součástí widgetu je i objekt datového zdroje, který má za úkol komunikaci mezi widgetem a zdrojem dat. Musí umět přijímat data ze vzdáleného zdroje, nebo ze statického JSON souboru. Dále musí získaná data převést do formátu vhodného pro časovou osu, která je následně zobrazí.

Pro realizaci widgetu autor využívá již existující knihovny. Konkrétně jQuery a jQueryUI. Obě knihovny ulehčují práci programátora již hotovými řešeními pro jazyk JavaScript. Knihovna jQueryUI je rozšířením první knihovny, která obsahuje navíc nástroje pro práci s grafickým rozhraním. Další knihovnou je RequirejS, jejímž úkolem je načítání skriptů až ve chvíli, kdy jsou potřeba, čímž urychluje načítání stránek. Moment.js slouží pro práci s časem, řeší mnoho problémů spjatých s formátem data a jeho manipulací. Nástroj pro vykreslení vztahů mezi událostmi se jmenuje Snap.svg [7]. Ten umožňuje práci se SVG na vysoké úrovni. Omezení je pouze to, že funguje na moderních prohlížečích s podporou nejnovějších funkcí. Pro další práci autor vytvořil několik svých podpůrných modulů:

- **oop.js** - Podpůrný modul maskující odlišný přístup JavaScriptu k vytváření tříd. Zavádí typ Closure, který uchovává kód funkce společně s daty nebo prostředím [11, str. 49]. Dále zavádí typ Class, díky němuž se dá v JavaScriptu

vytvořit třída stejně jako například v Javě. Další vlastností je i možnost dědění [11, str. 50]. Díky tomuto modulu lze definovat i rozhraní. V podstatě jsou to jednoduché objekty s prázdnými třídami [11, str. 52].

- **moment.extension.js** - Toto rozšíření již existující výše zmíněné knihovny doplňuje dodatečné funkce z oblasti formátování času pro potřeby osy [11, str. 52].

Aplikace je rozdělena do logických celků v rámci adresářové struktury. Ve složce css jsou uloženy styly pro časovou osu. Ve složce data je uložen skript pracující s daty. Ve složce auxiliary jsou veškeré obslužné skripty aplikace, ve složce lib jsou knihovny třetích stran a ve složce cz jsou hlavní soubory pro práci s daty, obecné třídy a rozhraní, pásy a položky pásů a rendery a pomocné třídy. Soubor index.html je potom v nejvyšší vrstvě [11, obr. 8.2, str. 53]. Na straně [11, str. 54] začíná stručný popis jednotlivých tříd a rozhraní v rámci widgetu.

5 Návrh aplikace

Základní částí aplikace je widget pro časovou osu, kterou vytvořil pan Kacerovský. Aplikace samotná se bude sestávat z více částí, které obstarávají funkcionalitu podle zadání:

- Správa uživatelů
- Vytváření vlastní časové osy
- Vytváření testů
- Realizace testů

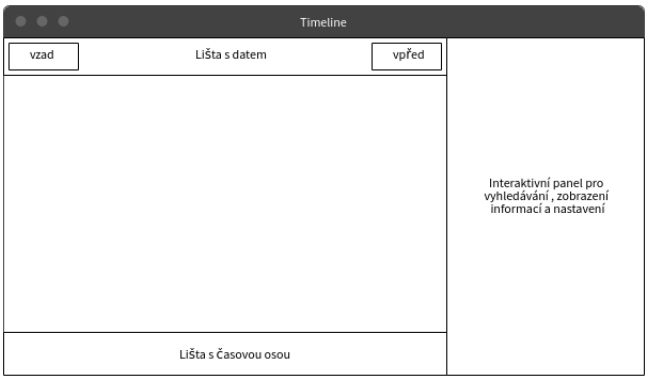
Z výše uvedených dat vyplývá, že aplikace bude muset být rozdělena na dvě části. První část, uživatelská, bude fungovat jako jednoduchá prezentace časových os a testů, případně bude obsahovat základní informace. Nejdůležitější součástí bude správné začlenení widgetu.

5.1 Uživatelská část

Uživatelská část by měla fungovat jako jednoduché rozhraní, které nabídne uživateli seznam časových os, které může procházet, a seznam všech přístupných testů, které může vyplnit. Součástí této části bude i jednoduchý formulář, pomocí kterého budou moci uživatelé vyjádřit svoje názory a připomínky. Protože některé osy mohou být rozpracované a neúplné, budou mít administrátoři možnost vybrat osy, které bude moci uživatel procházet. Zároveň by nemělo být možné nad neaktivní časovou osou spustit test. Seznam časových os bude reprezentován tabulkou, která bude obsahovat vždy jméno časové osy a tlačíko pro její procházení. Seznam dostupných testů bude mít stejnou podobu, navíc však bude nabízet vygenerování URL odkazu pro jednoduché sdílení.

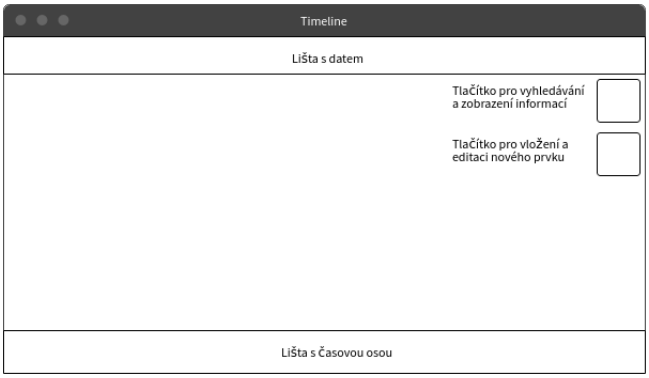
5.1.1 Widget

Ve svém základním provedení widget demonstruje veškerou funkcionalitu. Díky tomu může působit poněkud nepřehledně. Pro účel této práce je widget stěžejní a proto bude nutné jeho prostředí odlehčit. Widget také musí reflektovat jednotlivé módy zobrazení (procházení dat, vytváření testů a os). Ty aplikace rozezná pomocí URL parametrů.



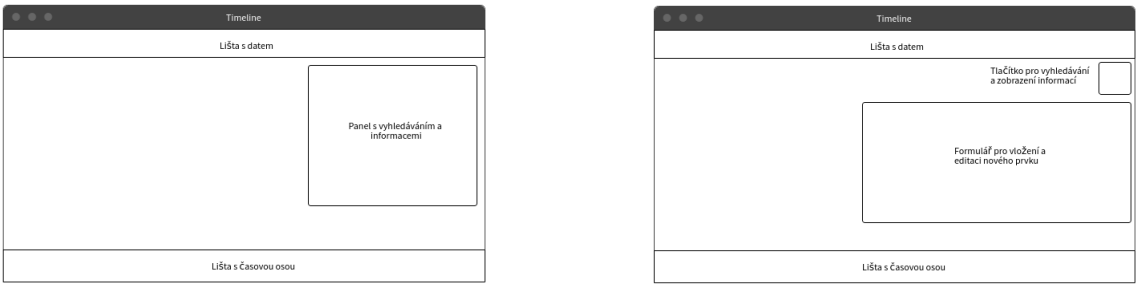
Obr. 4: Původní podoba widgetu

Aby byla vidět co největší část časové osy a minimalizovala se tak možnost překrytí dat, bude nejlepší využít zvětšování a zmenšování jednotlivých prvků podle toho, zda je uživatel využívá. Z původní podoby widgetu tedy zmizí Interaktivní panel v pravé části. Protože se ukázalo, že nejlepší metoda manipulace s časovou osou je pomocí myši, budou odebrána také tlačítka pro přesouvání se po ose vpřed a vzad. Místo Interaktivního panelu přidám dvě tlačítka reprezentující vyhledávání a přidávání/editaci prvku časové osy.



Obr. 5: Nová podoba widgetu

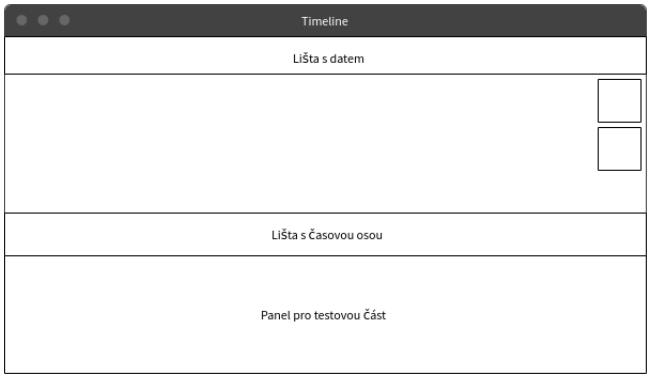
Tato tlačítka jsou interaktivní, a při kliknutí myší se zvětší a ve vzniklé oblasti se objeví příslušné ovládací prvky. Uživatel si tak může sám rozhodnout, kdy potřebuje spíše vidět na osu a kdy potřebuje něco vyhledat nebo přidat. Tlačítko pro přidávání a editaci je také přístupné pouze lidem přihlášeným.



(a) Po kliknutí na vyhledávací prvek (b) Po kliknutí na vkládací/editční prvek

Obr. 6: Widget s novými interaktivními prvky

Další funkcionalita widgetu spočívá v tvorbě a vyplňování testů. Uživatel musí mít přístup k panelu, který bude obsahovat formulář pro vytváření testových otázek nebo formulář pro odpovídání na tyto otázky. Zároveň by uživatel měl mít přehled, aby mohl hledat údaje pro vytváření nebo vyplňování otázek. V případě tvorby testu bude také dobré poskytnout uživateli seznam všech doposud vytvořených otázek, aby je mohl v případě potřeby upravit. Aby se minimalizoval dopad na přehled v rámci dat, bude nejlepší zúžit osu ve vertikálním směru. Díky tomu zůstane osa v hlavním horizontálním směru nezmenšená, což umožní vidět co největší část dat spolu se souvislostmi.



Obr. 7: Návrh pro tvobu a vyplňování testu

5.2 Administrativní část

Administrativní část musí být chráněna před možným přístupem uživatelů, kteří by neměli mít možnost manipulovat s daty v databázi. Za tímto účelem je vstup do administrace povolen pouze přihlášeným uživatelům. Ti budou moci získat přístup jen pokud jim administrátor zašle pozvánku na email, která obsahuje registrační odkaz. Díky tomu bude zajištěn přehled o všech uživatelích, kteří mají možnost manipulovat s daty. Administrátor bude také moci využít nástroje uživatelských rolí, díky němuž může ostatním registrovaným uživatelům povolit pouze určité akce. V rámci

administrativní části bude možné vytvářet vlastní časové osy. Uživatel tak naplní časovou osu daty podle potřeby a uloží do databáze. Samozřejmostí je možnost pozdějšího editování nebo smazání takto vytvořené časové osy. Vytváření testů zajistí další sekce, ve které bude uživatel vytvářet testy k existujícím časovým osám. Aby se zamezilo nehodám, test půjde smazat pouze autorovi nebo administrátorovi s vyšší rolí. Další částí administrace bude sekce pro vyhodnocování testů. Jedná se o stěžejní část celé aplikace, kde je možno procházet statistiky testů, které podstoupili uživatelé.

6 Implementace

Jádro aplikace tvoří PHP framewok Laravel spolu s MySQL databází. Aby mohl widget poskytovat popsanou funkcionalitu, bylo nutné ho v základu rozšířit o třídu `ButtonsPanel`. Obsluha testů a časové osy byla implementována pomocí scriptů v jazyce JavaScript [14]. Jedná se o soubory `ui.js`, `buttons.js` a `test-panel.js`.

6.1 Databáze

Databáze musi být koncipována tak, aby udržela data v přehledné a konzistentní podobě. Databáze je strukturně jednoduchá a zároveň se nepředpokládají složité dotazy, proto bude pro aplikaci stačit databáze MySQL.

6.1.1 Databázový model

Datový model je znázorněn na Obr. 15 v příloze bakalářské práce.

6.1.2 Tabulka users

Tabulka users obsahuje informace o všech uživatelích - aktivních i pozvaných. Díky bezpečnostním prvkům také zamezuje zneužití pro proniknutí do systému. Protože aplikace pracuje při přihlášení s emailem a nikoliv s uživatelským jménem, je této skutečnosti využito. Vždy při zadání emailu, na který je odeslána pozvánka, je vytvořen nový uživatel, který je zároveň označen jako neaktivní. Tabulka obsahuje následující údaje:

- **id** - unikátní identifikátor uživatele
- **role_id** - identifikátor role uživatele z tabulky ***roles***
- **email** - email uživatele používaný k přihlášení
- **password** - zahashované heslo uživatele
- **status** - indikátor toho, zda je uživatel aktivován a odpověděl na pozvánku.
- **remember_token** - ochrana proti zneužití "remember me" cookie
- **created_at** - datum vytvoření uživatele
- **updated_at** - datum aktualizace uživatele

6.1.3 Tabulka tests

Všechny vytvořené testy jsou uchovávané v této tabulce spolu s odkazy na uživatele a časovou osu. Data jsou ve formátu JSON. Důvodem této volby navázání na práci pana Kacerovského je, že widget pro zobrazení časové osy pracuje s daty právě v tomto formátu. Stěžejním bodem práce jsou testy samotné. Ukládání dat do databáze na úrovni uzlů a relací mezi nimi bylo předmětem jiné práce. V tabulce test jsou uloženy následující informace:

- **id** - unikátní identifikátor testu
- **user_id** - identifikátor autora testu
- **timeline_id** - identifikátor časové osy, k níž se test vztahuje
- **name** - jméno testu
- **data** - data testu ve formátu JSON
- **visible** - indikuje, zda je test viditelný při výběru k testování
- **created_at** - datum vytvoření testu
- **updated_at** - datum aktualizace testu

6.1.4 Tabulka timelines

Data časových os jsou ze stejného důvodu jako u dat testových ve formátu JSON. Pomocí spojení s tabulkou uživatelů se dá určit autor a díky tomu také zamezit ostatním uživatelům, kromě výše postavených administrátorů, aby časovou osu neměnili nebo dokonce nesmazali. Tabulka obsahuje následující informace:

- **id** - unikátní identifikátor časové osy
- **user_id** - identifikátor autora časové osy
- **name** - jméno časové osy
- **data** - data časové osy ve formátu JSON
- **visible** - indikuje, zda je osa viditelá k procházení
- **created_at** - datum vytvoření časové osy
- **updated_at** - datum aktualizace časové osy

6.1.5 Tabulka invitation

Všechny pozvánky se z bezpečnostních důvodů uchovávají, aby v případě využití pozvánky mohlo být ověřeno, že taková pozvánka byla opravdu vytvořena a uživatel tak mohl dokončit svou registraci vyplněním hesla. Všechny pozvánky jsou pouze jednorázové, což indikuje pole *used*. Data o pozvánkách jsou do tabulky ukládány v následujících sloupcích:

- **id** - unikátní identifikátor pozvánky
- **user_id** - identifikátor pozvaného uživatele
- **token** - token pro kontrolu při uplatnění pozvánky uživatelem
- **used** - indikátor použité pozvánky
- **created_at** - datum vytvoření pozvánky
- **updated_at** - datum aktualizace pozvánky

6.1.6 Tabulka roles

Tato tabulka obsahuje seznam všech rolí. Odkazy na role mají uživatelé v tabulce *users*. Ke každé roli se váží v aplikaci určitá oprávnění, která zamezují nebo povolují uživatelům přistupovat do různých sekcí nebo provádět rozličné akce. Tabulka obsahuje následující informace:

- **id** - unikátní identifikátor role
- **name** - jméno role
- **created_at** - datum vytvoření role
- **updated_at** - datum aktualizace role

6.1.7 Tabulka test_actions

Účelem této tabulky je uchovávat data jednotlivých testů, které uživatelé spustí a začnou vyplňovat. Tabulka také rozlišuje testy dokončené a nedokončené, čímž se zajistí konzistence statistik. Tabulka obsahuje následující informace:

- **id** - unikátní identifikátor testové akce
- **test_id** - identifikátor testu, který uživatel vyplňuje
- **participant_id** - identifikátor osoby, která test vyplňuje
- **started** - čas začátku testu

- **ended** - indikátor, zda byl test kompletně vyplňen
- **creatd_at** - čas vytvoření položky v databázi
- **updated_at** - čas aktualizace položky v databázi

6.1.8 Tabulka participants

Tabulka uchovává data jednotlivých osob, které začaly vyplňovat nějaký test. Data slouží hlavně ke statistickým účelům, protože obsahují určité informace o osobě. Tabulka obsahuje následující informace:

- **id** - unikátní identifikátor účasníka testu
- **email** - email účastníka testu
- **age** - věk účastníka testu
- **education** - vzdělání účastníka testu
- **work** - údaj o zaměstnání/studiu účastníka testu
- **created_at** - čas vytvoření položky účastníka testu v databázi
- **updated_at** - čas úpravy položky účastníka testu v databázi

6.1.9 Tabulka answers

Tato tabulka uchovává jednotlivé odpovědi účastníků testů. Díky odkazu na testovou akci je tak možné lehce získat odpovědi k jednotlivým testům a na jejich základě vytvářet požadované statistiky. Tabulka obsahuje následující informace:

- **id** - unikátní identifikátor odpovědi
- **test_action_id** - identifikátor testové akce, k níž se odpověď vztahuje
- **number** - číslo odpovědi na danou otázku
- **question_id** - indentifikátor otázky, na kterou účastník testu odpovídání
- **created_at** - čas vytvoření položky odpovědi v databázi
- **updated_at** - čas úpravy položky odpovědi v databázi

Tabulky password_resets a migrations jsou automaticky vytvářené frameworkem kvůli vnitřní funkcionalitě aplikace a nemají z hlediska této bakalářské práce větší význam.

6.2 Klient

Klientská část aplikace představuje co možná nejpřehlednější reprezentaci všech nástrojů a možností přístupných uživatelům. Obsahuje seznam přístupných časových os, které mohou uživatelé procházet. Dále obsahuje seznam testů, které si může kdokoliv zkusit vyplnit. Hlavní část klientské části aplikace představuje widget pro časovou osu, který je doplněný o funkce přístupné obyčejným uživatelům. Mezi tyto funkce patří možnost procházení všech časových os včetně prohledávání dat a možnost vyplnění testu na základě určité časové osy. Uživatelé mohou také napsat a odeslat zpětnou vazbu a reakce na webovou aplikaci.

V případě této práce se klientská část bere jako ta část aplikace, kam mají přístup nepřihlášení uživatelé. Všechny takto přístupné stránky jsou připravené pomocí šablonovacího systému frameworku Laravel. Takto vygenerované stránky poskytují všechnu základní funkcionalitu. Obsluha widgetu, který je také vykreslen pomocí šablony, je však vytvořena pomocí JavaScriptu a JQuery knihovny. Zvolil jsem způsob vytváření globálních funkcí, které jsou díky tomu přístupné ve všech scriptech. Tyto funkce jsou navěšeny na příslušné objekty widgetu. Na **Kód 1** vidíme ukázkovou definici globální funkce. Klíčová je hlavička funkce, která díky $.fn. části definuje funkci jako globální. Jediná podmínka, která musí být splněna je, aby byly scripty k aplikaci připojeny ve správném pořadí, protože jinak by funkce nebyly dostupné ve scriptech připojených k aplikaci dříve, než script obsahující tyto funkce. Základní význam funkcí je popsán v komentářích zdrojového kódu. Zde popíši princip, na jakém fungují složitější funkce.

```
$.fn.function_name = function() {
    // Body of the function
};
```

Kód 1: Globální funkce

6.2.1 Třída Timeline

Tato třída je dílem pana Kacerovského a obsluhuje veškerou funkcionalitu časové osy. Spojuje dohromady komponenty, které vykreslují časovou osu jako jeden celek. Tuto třídu jsem využil jako spojovací článek mezi stávající a novou funkcionalitou. Napojil jsem na ní novou třídu `ButtonsPanel`. Díky tomu, že této nové třídě je v konstruktoru předána instance třídy `Timeline`, mohu nyní vcelku jednoduše manipulovat se zdrojem dat pro vykreslení na časové ose.

6.2.2 Třída ButtonsPanel

Třída je psána ve stejném duchu jako původní třídy widgetu. Jejím úkolem je vykreslit prvek pro vyhledávání v datech časové osy a prvek pro vkládání nových dat

do časové osy. Obsahuje formulář, který umožní přidat jeden nový uzel s možností napojení na ostatní uzly různými typy relací. Data z tohoto formuláře jsou následně pomocí AJAXu odeslána na server a zároveň s tím vykreslena do časové osy.

Hlavním prvkem třídy je funkce `_insertData()`. Ta získá z formuláře potřebné údaje, následně podle nich vytvoří novou instanci třídy `Entity`, která představuje uzel na časové ose. Dále pak projede všechny relace vytvořené pro teto uzel a vytvoří pro ně instance třídy `Relation`. Data jsou následně odeslána na server a přidána do zdroje dat časové osy, která je nakonec překreslena, aby reflektovala i nově vložená data.

Kód 2 získává zvlášť uzly a zvlášť hrany z formuláře pomocí dvou funkcí, které projdou pole formuláře a získají příslušná data.

```
var node = this._processLeftPart(left);
var edge = this._processRightPart(right, node);
```

Kód 2: Získání dat z formuláře

Kód 3 ukazuje využití instance třídy `Timeline` k získání zdroje dat pro časovou osu, který bude následně aktualizován o nově vložený prvek.

```
var dataSource = this._timeline.getDataSource();
```

Kód 3: Získání zdroje dat pro časovou osu

V **Kód 4** jsou získaná data z **Kód 2** ukládána jako instance tříd `Entity` a `Relation` představující uzel dat a k němu náležející relace. Následně jsou ukládány do zdroje dat získaného z **Kód 3**.

```
var entity = new Entity(node);
dataSource._entities._data.push(entity);
this._data.nodes.push(node);

if(edge !== null && edge !== undefined) {
    var relation = new Relation(edge);
    dataSource._relations._data.push(relation);
    dataSource = this._updateRelations(edge, dataSource);
    this._data.edges.push(edge);
}
```

Kód 4: Tvorba instancí Entity a Relations

Nakonec se v **Kód 5** data odešlou na server funkcí `_sendData()`. Dále se nastaví nový zdroj dat pro časovou osu a samotná osa se překreslí. Odesílání na server probíhá pomocí standardního AJAX volání, kdy se nejprve definuje typ požadavku, data, která posíláme, a typ těchto dat, který je v tomto případě JSON.

```
this._sendData();
this._timeline.setDataSource(dataSource);
this._timeline.redraw();
```

Kód 5: Odeslání, uložení a vykrelsení dat

V **Kód 6** se pro AJAX připraví CSRF token, který slouží k ochraně před CSRF útokem. Tento funguje na bázi odposlechu komunikace mezi uživatelem, který je přihlášený a serverem. Útočník se tak může vydávat za uživatele a provádět autorizované operace, za předpokladu, že dokáže odhadnout url adresy a parametry pro administrativní část aplikace. Právě CSRF token coby náhodně generovaný token, který je odesílán při každé komunikaci a generován při každém novém odesílání požadavků na server znovu, zabraňje CSRF útoku, protože útočník nedokáže nikdy vygenerovat token stejný jako aplikace. Pokud tak odešle starý token, aplikace tento request ignoruje.

```
$.ajaxPrefilter(function(options, originalOptions, xhr) {
    var token = $('meta[name="csrf-token"]').attr('content');

    if (token) {
        return xhr.setRequestHeader('X-CSRF-TOKEN', token);
    }
});
```

Kód 6: Připojení CSRF tokenu

V **Kód 7** se data upravují tím, že JSON objekty jsou převedeny na řetězce. Tyto řetězce pak mohou být jednoduše odeslány.

```
var nodes = JSON.stringify(this._data.nodes);
var edges = JSON.stringify(this._data.edges);
```

Kód 7: Upravení dat pro odeslání

V **Kód 8** se připravuje samotný AJAX požadavek na server. Nejprve se určí adresa, na kterou bude požadavek odeslán. Tato adresa se musí změnit podle toho, kde běží serverová část. Dále se definují data, která budou odeslána, typ požadavku a typ

dat. Na `jqxhr` objekt se mohou navěsit různé stavové funkce. Jejich úplný seznam lze nalézt na stránkách JQuery knihovny. Tyto funkce pomáhájí sledovat v jakém stavu se požadavek nachází. Například poznáme, kdy se odesílá, kdy byl odeslán a zda byl odeslán v pořádku. Pro účely aplikace postačí sledovat, zda požadavek proběhl či neproběhl úspěšně.

```
var jqxhr = $.ajax({
    url : 'http://127.0.0.1:8000/timelines/' + id,
    data : JSON.stringify({'nodes': nodes, 'edges': edges}),
    type : 'PATCH',
    contentType : 'application/json',
    processData: false,
    dataType: 'json'
});
```

Kód 8: Odeslání dat

Na objekt AJAX požadavku odesílaného na server je v **Kód 9** navěšena funkce pro detekci dokončeného odesílání a funkce detekce erroru.

```
jqxhr.done(function(data, textStatus, jqXHR) {
    console.log('Succeed:', textStatus);
});

jqxhr.fail(function(jqXHR, textStatus, errorThrown) {
    console.log('Failed:', errorThrown);
});
```

Kód 9: Definice funkcí pro sledování stavu požadavku

6.2.3 Soubor test-panel.js

Soubor test-panel.js obsahuje veškerou funkcionalitu pro vykreslení panelu pro tvorbu testů a také pro testy samotné. Nacházejí se v něm funkce, které uzpůsobí aplikaci tak, aby časová osa nebyla překryta testovým panelem. Dále obsahuje ovládací prvky pro tvorbu testových otázek, jako je přidávání různého počtu odpovědí, označení správné odpovědi a také validace formuláře. Při každé změně testových dat jsou data odeslána na server, aby byla práce v každém okamžiku zálohována. Komunikace se serverem probíhá pomocí AJAXu.

Testový panel je součástí šablony pro vykreslení widgetu. Jeho viditelnost je vždy odvozena od módu, ve kterém je widget spušťěn. Tento mód je vždy přítomen v URL

parametru **mode**. V základním zobrazení obsahuje panel dvě části. V levé části je samotný formulář, který obsahuje políčko pro otázku, jedno políčko pro odpověď spolu s zaškrtávacím políčkem pro označení správné odpovědi, tlačítko pro přidání odpovědi, tlačítko pro uložení otázky a tlačítko pro ukončení tvorby testu.

Funkce `addQuestion()` nejprve provede validaci formuláře a zkontroluje, zda jsou všechna pole vyplněná. Pokud nejsou pole vyplněná, funkce vrátí seznam původních otázek. Poté se proiterují vstupy pro odpovědi a uloží se do pole JSON objektů(viz **Kód 9**), ve kterém je u každé odpovědi uložena informace, zda je tato odpověď správná nebo špatná. Pomocí získaných dat se vytvoří objekt pro otázku(viz **Kód 10**), ve kterém je text otázky z políčka pro otázku, dále pole odpovědí a ID otázky.

```
var answer = {
    'text': answersString,
    'right': false/true
}
```

Kód 10: Objekt odpovědi

```
var newQuestion = {
    'id': id_otazky
    'question': questionColumn.val(),
    'answers': answers
}
```

Kód 11: Objekt otázky

Zároveň s těmito úkony je každá vložená otázka zobrazena v pravé části testovacího panelu. Pokud se na otázku klikne, vyplní se v levé části formulář údaji z této otázky a otázka se tak může editovat. Funkce `editQuestion()` funguje na stejném principu, jako funkce `addQuestion()`, navíc se pouze získají data z pole otázek, podle kterých se vyplní formulář. Při každém přidání otázky se seznam v pravé části panelu vymaže a znovu naplní již připravenými otázkami pomocí funkce `repopulateAnswerList()`. Ta se volá i v případě vymazání některé otázky. Tato otázka se vyjme z pole otázek a zároveň se provede aktualizace seznamu v pravé části. Vždy při přidání nebo editaci nebo vymazání otázky se nová data posílají vcelku na server, který je ukládá do databáze.

6.2.4 Soubor buttons.js

Tento soubor obsahuje funkce pro zvětšování a zmenšování prvků pro vyhledávání dat a pro přidávání dat do časové osy. Zajišťuje, aby v každou chvíli byl zvětšen pouze jeden z těcho prvků pro maximalizaci přehlednosti. Další jeho funkcí je přidávání a odebírání polí pro nové relace do formuláře pro vkládání dat do časové

osy.

Funkce `resizeButton()` obsahuje základní rozměry příslušných tlačítek. Podle id tlačítka rozezná, na které bylo kliknuto, a potom zavolá funkci `minimize()` resp. `maximize()`, která prvek zvětší nebo zmenší. Zároveň s těmito funkcemi je vždy zavolána funkce `minizeOthers()`, která zmenší všechny ostatní prvky, na které nebylo kliknuto, a které jsou maximalizované. Díky tomu je vždy maximalizován pouze jeden prvek. Další funkcionalitou tohoto scriptu je přidávání políček pro nové relace při přidávání nového prvku do časové osy. Zpracování formuláře probíhá ve třídě `ButtonsPanel`, proto se zde pouze přidává relace.

6.2.5 Soubor test.js

Má na starost vše kolem vyplňování testu. Protože stejně jako u tvorby testu i panel s vyplňváním testových otázek je v šabloně na pevno, odvíjí se jeho viditelnost podle módu. Při začátku testu nejprve získá data z formuláře, které uživatel vyplnil. Při odkliknutí toho formuláře se začíná počítat čas startu testu. Jako odpověď na odeslaná data přijde obsah testu. Aby nemohl uživatel podvádět, není v odpovědi serveru žádná informace o správnosti odpovědí, tato kontrola se děje až na konci testu na straně serveru. Tuto prvotní funkcionalitu obstarává funkce `initStartTestButton`, která se spustí ve chvíli kliknutí na tlačítko odesílající informace z formuláře. Další funkce, `initNextquestion()` zajišťuje načtení další otázky, zároveň odešle odpověď na předešlou otázku na server. Součástí je také validace, zda uživatel vybral nějaku možnost. Všechny požadavky a odpovědi jsou realizovány pomocí AJAXu.

6.2.6 Soubor ui.js

Tento script se dá považovat za hlavní zastřešující prvek nad ostatními soubory. Podle URL získá hodnotu módu zobrazení widgetu časové osy(viz **Kód 11**) a následně přes konstrukci `if-else` upraví viditelnost jednotlivých prvků widgetu. Dále navěšuje funkce připravené ostatními scripty na příslušné prvky widgetu. Kvůli tomu, že jsou prvky dynamicky generované, musíme použít funkce `on()`. Tato funkce z knihovny JQuery dokáže na rozdíl od funkce `click()` navěsit různé funkce i na dynamicky vytvářené prvky. V ukázce(viz **Kód 12**) vidíme, že pomocí identifikátoru navěšujeme funkci na celý dokument. Nicméně při akci kliknutí se najde dynamicky generovaný prvek a akce se provede v kontextu tohoto prvku.

```
jQuery(document).on('click', 'identifikator', function() {
  jQuery(this).funkce(); //Nyni je this tlacitko
});
```

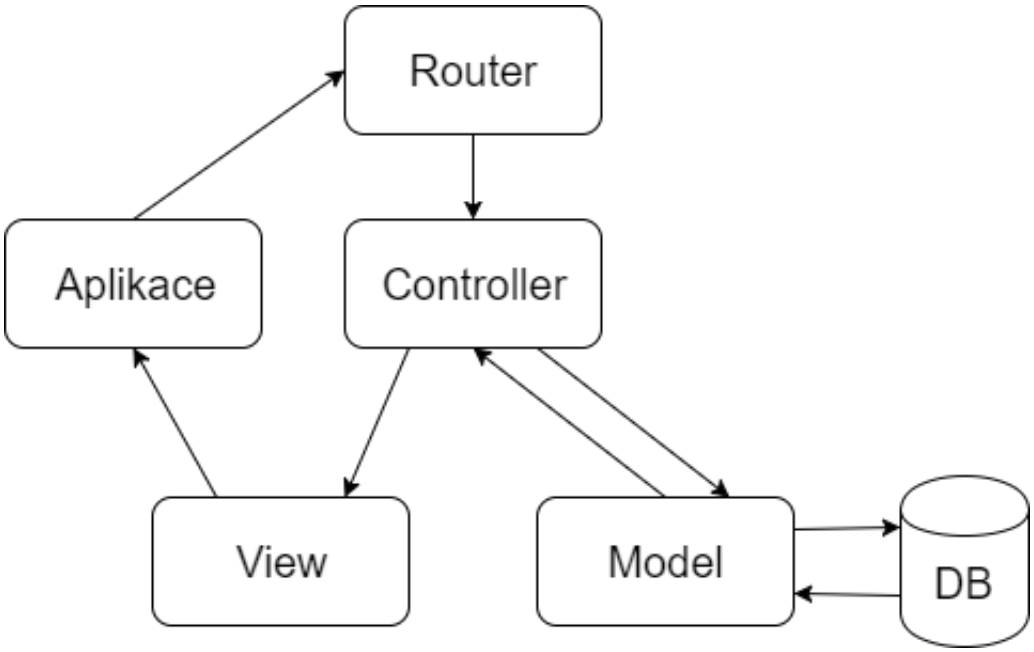
Kód 12: Ukázka použití funkce on()

6.3 Server

Serverová část je realizována pomocí PHP frameworku Laravel. Tento framewok umožňuje tvorbu rychlých a stabilních aplikací a aktuálně se řadí mezi nejlepší frameworky, což bylo hlavním důvodem k jeho výběru. Tento framework také dodržuje architekturu **M-V-C**(Model-View-Controller), což v praxi znamená, že aplikace je rozdělena do tří vrstev. Pvní vrstva **Model** má na starosti mapování databáze a práci s daty v ní. Další vrstva **View** je tvořena systémem šablon, které vykreslují aplikaci spolu s daty z další vrstvy **Controller**. Tato poslední vrstva má za úkol komunikaci s modelem a odesílání dat získaných v něm do šablon.

Mezi další výhody patří velmi silný nástroj pro příkazovou řádku, který umožňuje například procházení databáze a testování modelů aplikace, generování databázových tabulek spolu s vazbami a případně daty, generování tříd modelů a controllerů včetně předpřipravených základních funkcí. Veškeré informace o možnostech frameworku jsou dostupné v dokumentaci frameworku. Důležitou součástí framewoku je systém šablon, díky kterému se dá velmi jednoduše integrovat dynamický kód do HTML kontextu. Použití HTML a CSS pro stylování je u webových aplikací samozřejmostí.

Tento framework představuje celou webovou aplikaci. Obstarává přihlášení a také registrování uživatelů, dále ukládání dat do databáze a jejich poskytování widgetu pomocí jednoduchého API, zobrazení widgetu a možnost odeslání zpětné vazby. Dokumentace k tomuto framewoku je k dispozici na internetu, a proto shrnu jen důležité části aplikace.



Obr. 8: Životní cyklus aplikace [1]

Podle Obr. 8 životní cyklus aplikace začíná požadavkem vyvolaným aplikací, který je následně zpracován routerem. Podle požadavku router rozhodne, jaká funkce v Controlleru se spustí. V rámci funkce proběhne komunikace s modelovou vrstvou, která získá data z databáze. Ta jsou předána zpět funkci v controlleru a v případě potřeby upravena do požadovaného tvaru. Controller poskytne data šablonám, které vykreslí výsledek požadavku.

6.3.1 Soubor /routes/web.php

Tento soubor má na starost analyzování příchozího pořadavku, podle kterého určí, jaká funkce má být zavolána z jakého controlleru. Jsou zde také definovány routy, které jsou přístupné pouze přihlášeným uživatelům. V případě neautorizovaného přístupu je uživatel přesměrován na domovskou stránku.

6.3.2 Modelové třídy

Soubory těchto tříd se nacházejí v **/app/** složce. Třídy samotné tvoří vrstvu mezi databází a controllery a slouží pro práci s databází. Pokud se dodrží konvence pojmenovávání tabulek a těchto modelových tříd, dokáže framework jednoduše namapovat data v databázi na tyto třídy. Pokud je využita možnost definování vazeb mezi těmito entitami, dokáže framework při mapování dat zohlednit i tyto vazby, díky čemuž se jednoduše při vyhledání entity získají i data, která jsou s touto entitou v databázi spojena relací. Modelové třídy také umožňují velmi jednoduše konstruovat dotazy do databáze, aniž by bylo nutné psát samotné SQL dotazy [4].

Díky definici funkce `role()` modelová třída pro uživatele ví, že mezi uživatelem a rolí je vazba, a framework to může zohlednit při mapování dat. Pokud vyhledáme uživatele, můžeme použít `$user->role` i když žádné takové políčko v tabulce uživatelů přímo není.

6.3.3 Controllery

Nejdůležitější část frameworku tvoří controllery. Tyto třídy se nacházejí v **/app/http/Controllers/**. Každý controller by měl obsluhovat jednu logickou část aplikace. Tuto skutečnost reflektují i jména controllerů. Protože se jedná o aplikaci jednodušší a také tím, že logické celky aplikace jsou menší a je jich více, neobsahují controllery mnoho funkcionality. Ta je pokryta komentáři v rámci kódu aplikace.

V základu jsou tvořeny funkcemi pro zpracovávání požadavků, které jsou jim předány routerem. Podle typu požadavku funkce získá data z databáze nebo do databáze zapíše. Pokud jako návratovou hodnotu funkce použijeme čistě proměnnou obsahující data z databáze, aplikace z ní udělá JSON objekt. Tato funkcionalita se hodí například pro API, kdy klientská část potřebuje pouze data. V druhém případě

můžeme definovat, aby funkce vrátila odkaz na šablonu, která se má vykreslit spolu s daty, která se mají v šabloně použít.

Funkce `index()` v tomto případě vrací funkci `view()`, která pomocí parametru **test.index** vybere příslušnou šablonu. V rámci této šablony je k dispozici proměnná **$data**, kterou vytvoří funkce `compact()`.

Laravel framewrok má svůj šablonovací systém, díky kterému je jednoduché manipulovat s PHP proměnnými v rámci šablon. Mimo to tento systém nabízí další užitečné nástroje, jako je zanořování šablon, skládání šablon z menších částí atd. Nacházejí se ve složce **/resources/views/**. Uživatel si zde může navrhnout vlastní hierarchii složek a souborů. Ta se vždy zohlední v odkazu na šablonu pomocí tečkové notace. Soubor **test.index** můžeme tedy nalézt ve složce test pod názvem index.blade.php. Stejně jako mnoho ostaních frameworků i Laravel nabízí nástroje pro práci s PHP proměnnými v rámci šablon. Kompletní přehled můžeme nalézt na stránkách frameworku [2].

7 Testování

Jako metoda testování aplikace bylo vybráno uživatelské testování. Testeři postupovali podle připravených scénářů, které pokrývali aplikaci ve dvou částech. Zvlášť se testovaly úpravy provedené na widgetu, zvlášť administrativní část a zvlášť uživatelská část. Scénáře obsahovaly pokyny, kterými se testeři měli řídit. Zároveň byl ponechán prostor pro testery, aby ke svému hodnocení připojili i připomínky a postřehy. Aplikaci testovalo ve všech částech 8 testerů vždy pomocí prohlížečů Chrome, Edge a Firefox, čímž se otestovala i kompatibilita s jednotlivými prohlížeči. Během vytváření aplikace a během testů se ukázalo několik nedostatků, které mohou být předmětem dalšího rozšíření aplikace.

7.1 Testování uživatelské části

7.1.1 Scénář: Otevření časové osy pro prohlížení

Úkol: Na stránkách aplikace vyzkoušejte možnost otevření časové osy pro prohlížení dat.

Testování proběhlo bez problémů a všichni testeři se dostali na stránku s widgetem pro prohlížení časové osy. Někteří si stěžovali, že titulek osy není dostatečnou informací ohledně obsahu a uvítali by, aby osy měly i krátké popisy toho, co obsahují.

7.1.2 Scénář: Spuštění uživatelského testu

Úkol: Na stránkách aplikace spusťte test nejprve pomocí tlačítka Spustit a poté pomocí generovaného odkazu.

Toto testování bylo v pořádku pro obě metody otevření testu, cílem byla kontrola správného generování odkazu jak na tlačítku, tak v možnosti vygenerování odkazu pro sdílení. Pro sekci testů by testeři také uvítali krátký popisek testu, nebo alespoň jméno osy, ke které se test vztahuje.

7.2 Testování administrativní části

7.2.1 Scénář: Testování administrativní části

Úkol: Pomocí přihlašovacích údajů poskytnutých autorem aplikace se přihlašte do administrace. Následně si pošlete pozvánku a vyzkoušejte registraci pomocí odkazu v pozvánce. Odhlašte se a přihlašte se znovu pomocí svých přihlašovacích údajů. Po přihlášení se odhlašte a pokuste se přistoupit na URL poskytnutou autorem.

Přihlášení do aplikace proběhlo pomocí defaultních údajů bez problémů. Doručení odeslané pozvánky proběhlo ve všech případech takřka okamžitě. Registrace do systému pomocí odkazu byla bez komplikací. V případě přístupu na poskytnutou

url bez přihlášení aplikace přesměrovala testera na domovskou adresu. Testeři v žádném bodě scénáře neshledali problém. Jeden uvedl, že by uvítal, aby po odeslání pozvánky aplikace zobrazila informaci o tom, zda byla pozvánka odeslána nebo ne.

7.2.2 Scénář: Testování práv podle role

Úkol: Pomocí vlastních přihlašovacích údajů se přihlašte do aplikace. Vyzkoušejte možnost smazání uživatelů, testů a časových os ze systému. Následně se přihlašte pomocí údajů poskytnutých autorem aplikace a celý postup zopakujte.

Při přihlášení pomocí svých údajů testeři správně hlásili, že nemohou smazat žádnou osu ani test, který sami nevytvořili. Protože testeři nemají nejvyšší roli, nemohli smazat žádného uživatele ze systému. Po přihlášení s mnou poskytnutými údaji získali uživatelé nejvyšší možnou roli a tudíž mohli mazat všechny role, testy uživatele bez problému. Testerům se zamlouvalo dvoufázové mazání, kdy se aplikace nejprve zeptá, zda jsme si jistí, a až potom smaže vybraná data.

7.3 Testování widgetu

7.3.1 Scénář: Vytvoření časové osy

Úkol: Pomocí vlastních přihlašovacích údajů se přihlašte do aplikace. Vytvořte časovou osu a následně z domovské stránky vyzkoušejte, zda jde otevřít a zda obsahuje vámi vytvořená data.

Žádný tester nezaznamenal problém s vytvořením časové osy ani s jejím následným otevřením pro prohlížení. Testeři ocenili přehlednost widgetu a líbilo se jim zpracování formuláře pro přidání nového uzlu do časové osy. Během vkládání dat nebyl zaznamenán žádný problém. Během testování prohlížení dat v časové ose testeři zaznamenali, že prohlížeč Edge nevykresluje animace osy plynule, jako ostatní prohlížeče.

7.3.2 Scénář: Vytvoření testu

Úkol: Pomocí vlastních přihlašovacích údajů se přihlašte do aplikace. Vytvořte test a z domovské stránky vyzkoušejte, zda jde spustit a zda jsou otázky i odpovědi správně zobrazené.

Během tohoto testování se zároveň otestovala i funkčnost vyplnění testu. Testeři neměli s tvorbou testu problém a líbila se jim funkcionalita spojená s editací již vytvořených otázek. Formulář jim přišel přehledný. Testeři zvlášť ocenili možnost, že i během tvorby testu mohou editovat samotnou časovou osu a také do ní přidávat data. Nově vytvořený test byl plně funkční. Připomínky testerů směřovaly k době

potřebné pro zobrazení výsledků. Tento defekt se bohužel nedá nijak ze strany aplikace vyřešit, protože souvisí s tím, kde je aplikace hostována.

7.3.3 Závěr testování

Testování neodhalilo žádné skryté vady ani nefunkční část aplikace. Testeři hodnotili zpracování kladně až a několik vítek. Některé byly zohledněny v implementaci a jiné vzhledem k povaze a cíli práce mohou být předmětem budoucího rozšíření.

7.4 Nedostatky a možnosti rozšíření

Protože aplikace se zaměřuje na rozšíření widgetu a vytvoření funkcionality pro testování a vytváření nových časových os, může se aplikace v dalších aspektech jevit neúplně. Během testování se projevily také některé nedostatky, které může být žádoucí dodělat do budoucna.

7.4.1 Nedostatky aplikace

Mezi nedostatky aplikace patří to, že neinformuje uživatele o výsledcích akcí. U webových aplikací se většinou vyrábějí infoboxy, které vždy uživatele zpraví o výsledku. Tuto funkcionalitu jsem v rámci aplikace nezavedl ze dvou důvodů. Prvním důvodem byla podle mého názoru myšlenka, že tato funkcionalita je pro účely této práce nadbytečná, a bylo třeba zaměřit se na hlavní body práce. Druhým důvodem bylo to, že většina akcí, které uživatel provede, se projeví viditelně v aplikaci, čímž může uživatel poznat, že akce proběhla úspěšně.

7.4.2 Možnosti rozšíření

Aplikace zachycuje tvorbu časových os a testů v základní podobě. V rámci rozšíření by bylo možné ptát se uživatele na začátku testu na více věcí, aby byla statistická data úplnější. Mezi další rozšíření by mohla patřit možnost různých druhů testů, protože aplikace v základu umožňuje testy pouze zaškrtávací. Aplikace by se také dala rozšířit tím způsobem, že by uživatelé měli své účty a viděli by jak statistiky vlastního testu, tak statistiky v porovnání s výsledky ostatních lidí.

8 Závěr

Cílem této práce bylo vytvoření nástroje pro demonstraci využití časové osy, která byla vytvořena na Katedře informatiky a výpočetní techniky. V rámci této demonstrace vznikla aplikace umožňující vytváření časových os, které se dají procházet, dále testů podle těchto časových os. Testy je možné vyplňovat a díky tomu i určit, zda je časová osa užitečná při učení.

Pro realizaci aplikace byly vybrány zavedené technologie pro tvorbu webových aplikací. Díky výběru frameworku Laravel byla tvorba některých částí aplikace vcelku rychlá. Tento framework zároveň zaručuje zaručuje rychlost a stabilitu.

Výsledkem práce je webová aplikace s jednoduchým prostředím, které umožňuje uživatelům procházet jednotlivé časové osy a vyplňovat přístupné testy. V rámci administrace existuje možnost vytváření časových os a testů. Aplikace také umožňuje jednoduché statistické porovnávání výsledků. V rámci aplikace je kladen důraz na jednoduchost a přehlednost. Díky tomu je intuitivní a práce s ní nepředstavuje žádný problém.

Použitá literatura

[1] Laravel architecture.
URL http://laravelbook.com/laravel-architecture/

[2] Laravel Blade. 2017.
URL https://laravel.com/docs/5.4/blade

[3] Laravel command line tool. 2017.
URL https://laravel.com/docs/5.4/artisan#introduction

[4] Laravel Eloquent Model Retrieving. 2017.
URL https://laravel.com/docs/5.4/eloquent#retrieving-models

[5] Laravel trending. 2017.
URL https://trends.google.com/trends/explore?q=%2Fm%2F0jwy148,%2Fm%2F09cjcl,%2Fm%2F04n23m7,%2Fm%2F0cdvjh

[6] MySQL vs. PostgreSQL. 2017.
URL https://goo.gl/JNFe2f

[7] Snap.svg. 2017.

[8] Tiobe index. mar 2017.
URL https://www.tiobe.com/tiobe-index/

[9] Hessová, G.: *Automatické získání historických údajů z webových zdrojů*. 2015.

[10] Hrbáček, D.: *Zpracovnání časových údajů pro jejich vizualizaci*. 2015.

[11] Kacerovský, M.: *Vizuální reprezentace precedenčního grafu*. 2015.

[12] Merunko, D.: *Generování a vizualizace časové osy*. 2014.

[13] Nutile, A.: *Laravel 5.x Cookbook*. 2016, ISBN 978-1786462084.

[14] Žára, O.: *JavaScript*. 2015.

Slovník pojmů

AJAX Asynchronous JavaScript and XML, technologie umožňující asynchronně překreslovat aplikaci bez obnovení stránky a také získávat a odesílat data na server.

API Application Programming Interface - aplikační rozhraní, které definuje komunikaci s danou aplikací.

CSRF Cross Site Request Forgery, zneužití komunikace mezi uživatelem a serverem ve prospěch útočníka za účelem změny stavu dat.

CSS Cascading Style Sheets, jazyk popisující jakým způsobem se prvky webové stránky zobrazí.

framewok Struktura pomocných knihoven, nástrojů a návrhových vzorů.

HTML HyperText Markup Language, značkovací jazyk pro tvorbu statických webových stránek.

JSON JavaScript Object Notation, způsob zápisu dat nezávislý na platformě.

M-V-C Architektura pro tvorbu moderních dynamických webových aplikací.

MySQL Structured Query Language, jazyk využívaný pro vytváření dotazů do databáze.

PHP Hypertext Preprocessor, původně Personal Home Page - scriptovací jazyk využívaný pro tvorbu webových aplikací.

URL Uniform Resource Locator, řetězec znaků s danou strukturou. Slouží k přesné specifikaci umístění zdrojů na internetu.

A Uživatelská dokumentace

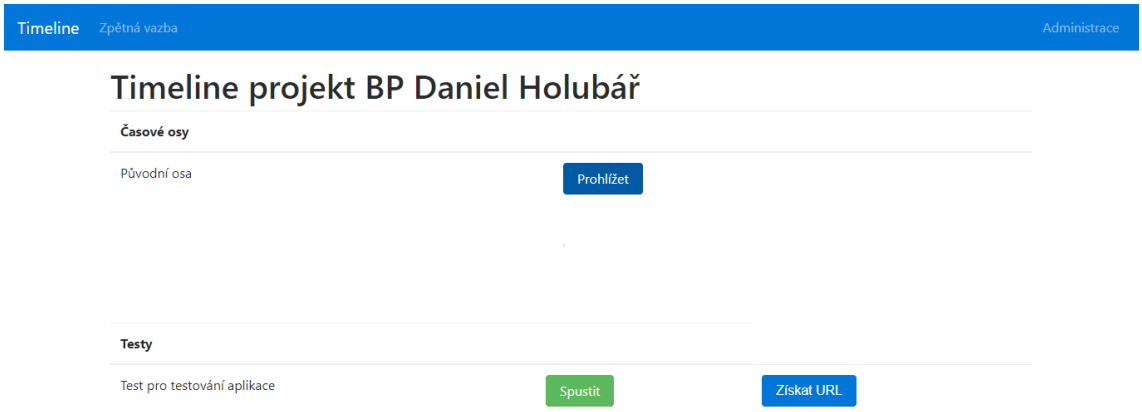
Následující sekce obsahuje návod na instalaci aplikace a její základní použití.

A.1 Systémové požadavky

Aby aplikace fungovala, musejí být splněny následující požadavky: [1]

- **MySQL 5.7**
- **PHP 5.6.4**
- **OpenSSL PHP rozšíření**
- **PDO PHP rozšíření**
- **Mbstring PHP rozšíření**
- **Tokenizer PHP rozšíření**
- **XML PHP rozšíření**
- **Apache 2.4**

A.2 Uživatelská část



Obr. 9: Výchozí obrazovka uživatelské části

[1] Podrobný návod k instalaci a spuštění aplikace je součástí CD

Uživatelská část nabízí několik možností interakce:

- Tlačítko **Prohlížet** v tabulce časových os
- Tlačítko **Získat URL** v tabulce testů
- Tlačítko **Spustit** v tabulce testů
- Odkaz na zpětnou vazbu v horní navigační liště
- Odkaz na administraci v horní navigační liště

Pokud je kliknuto na tlačítko **Prohlížet**, aplikace se přesměruje na widget s daty příslušné časové osy. Při kliknutí na tlačítko **Získat URL** se objeví okno s odkazem na příslušný test v podobě textového řetězce, který se dá jednoduše zkopírovat. Tlačíko **Spustit** spustí příslušný test. Při kliknutí na odkaz **Zpětná vazba** je uživatel přesměrován na stránku obsahující jednoduchý formulář, do kterého uživatel může napsat připomínky a poznámky k aplikaci. Pokud uživatel klikne na odkaz **Administrace**, tak v případě, že není přihlášen je přesměrován na obrazovku pro přihlášení uživatele. V opačném případě je přesměrován na obrazovku administrativní části aplikace.

A.3 Administrativní část

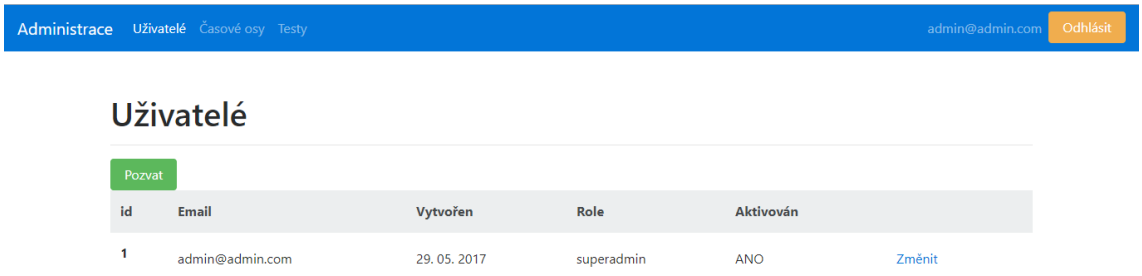
Obr. 10: Výchozí obrazovka administrativní části

Administrativní část nabízí následující možnosti:

- Odkaz na sekci uživatelů v horní navigační liště
- Odkaz na sekci testů v horní navigační liště
- Odkaz na sekci časových os v horní navigační liště

A.3.1 Správa uživatelů

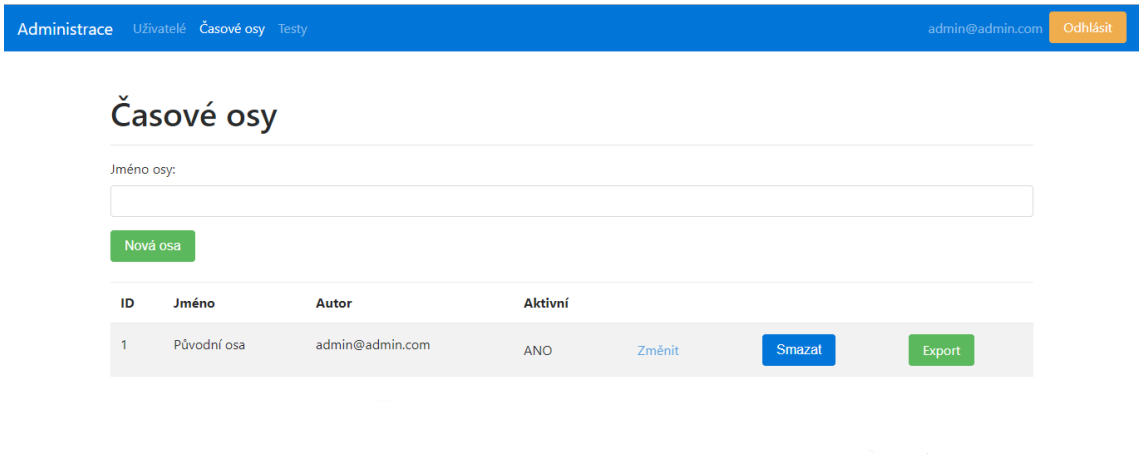
V této sekci je tabulka se všemi uživateli včetně těch, kterým byla teprve poslána pozvánka. Každému uživateli lze změnit role, případně lze uživatele smazat. K tomuto účelu slouží příslušná tlačítka. Do této sekce má přístup pouze uživatel s nejvyšší rolí. Po kliknutí na tlačítko pozvat se objeví obrazovka s jediným textovým polem pro email uživatele. Po jeho zadání se pošle email s vygenerovaným odkazem pro registraci uživatele.



Obr. 11: Obrazovka správy uživatelů

A.3.2 Správa časových os

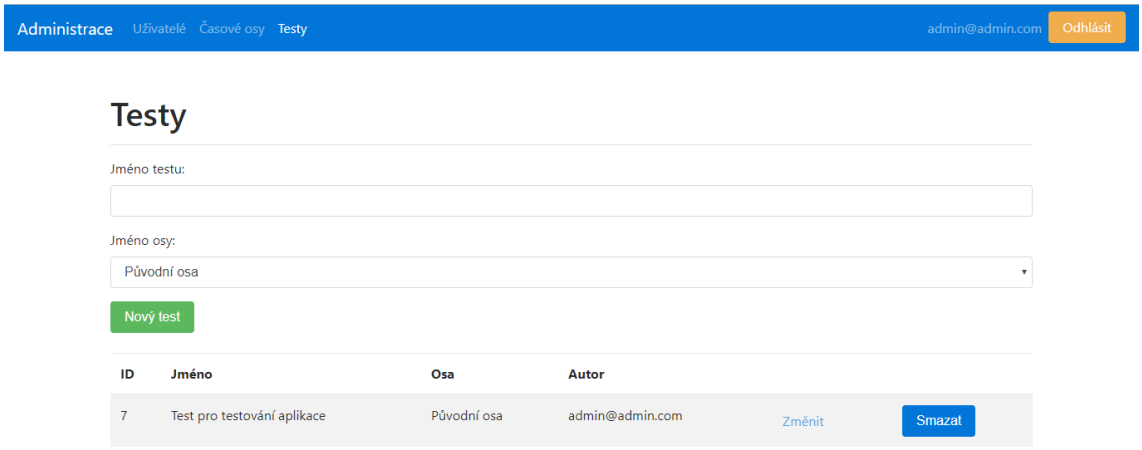
Sekce časových os ukazuje přehlednou tabulku se všemi časovými osami a základními údaji o těchto osách. Každá časová osa se dá smazat, i když všechny může mazat pouze superadmin, jinak může každou osu smazat pouze autor. Tlačítko export slouží k získání JSON souboru s daty časové osy. V horní části obrazovky je textové pole pro název nové osy spolu s potvrzovacím tlačítkem. Po zadání názvu je uživatel přesměrován na widget v módu tvorby nové osy.



Obr. 12: Obrazovka správy časových os

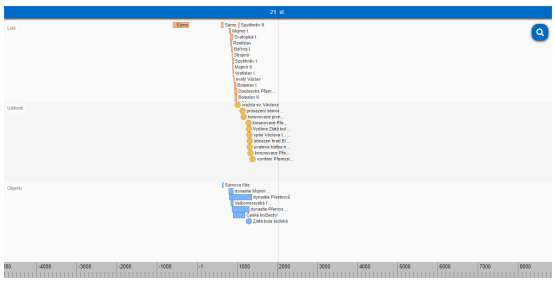
A.3.3 Správa testů

V sekci testů je také tabulka se všemi testy a základními údaji. Test je možné editovat případně smazat. Platí stejné pravidlo, jako u časových os. Všechny testy může mazat a editovat pouze superadmin. Normální uživatel může smazat nebo editovat pouze svůj vlastní test. V horní části obrazovky je formulář pro nový test. Po vyplnění názvu a přiřazení časové osy je autor přesměrován na obrazovku widgetu v módu vytváření nového testu.

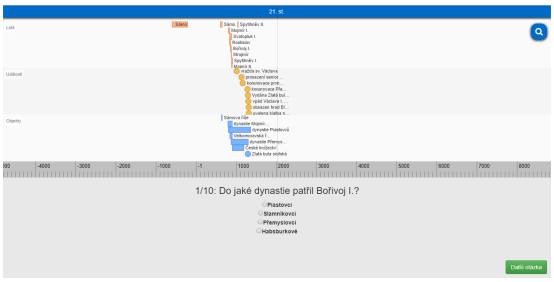


Obr. 13: Obrazovka správy testů

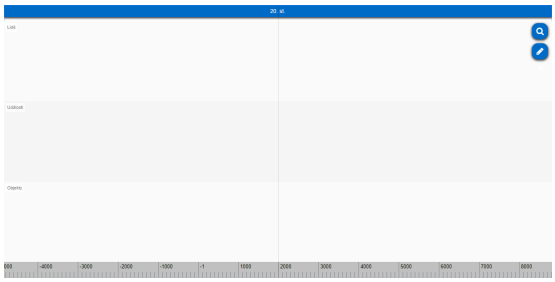
A.4 Widget



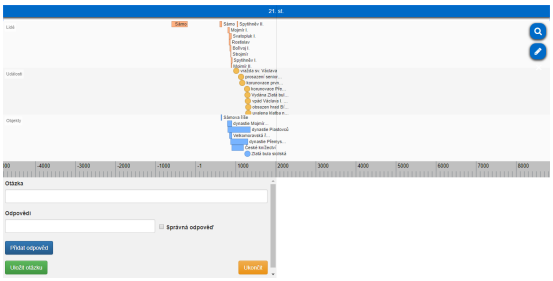
(a) Mód procházení dat



(b) Mód testu



(c) Mód vytváření osy



(d) Mód vytváření testu

Obr. 14: Módy widgetu

Widget nabízí několik módů(viz Obr. 14) podle toho, co se od něj požaduje:

- Tvorba testu - V tomto módu widget nabízí vyhledávácí nástroj, dále nástroj pro vkládání nových dat do osy a jejich editaci. Oba se nacházejí v pravé horní části obrazovky a po kliknutí na ně nabízejí buď políčko pro vyhledávání, nebo formulář pro nový uzel a relace k němu náležící. Hlavní částí je potom panel s formulářem pro tvorbu testových otázek. Formulář se nachází v levé části a umožňuje k jedné otázce připojit neomezeně odpovědí. Po uložení otázky se v pravé části tato otázka objeví v seznamu, kde je možné jí smazat nebo kliknout na ní pro editaci.

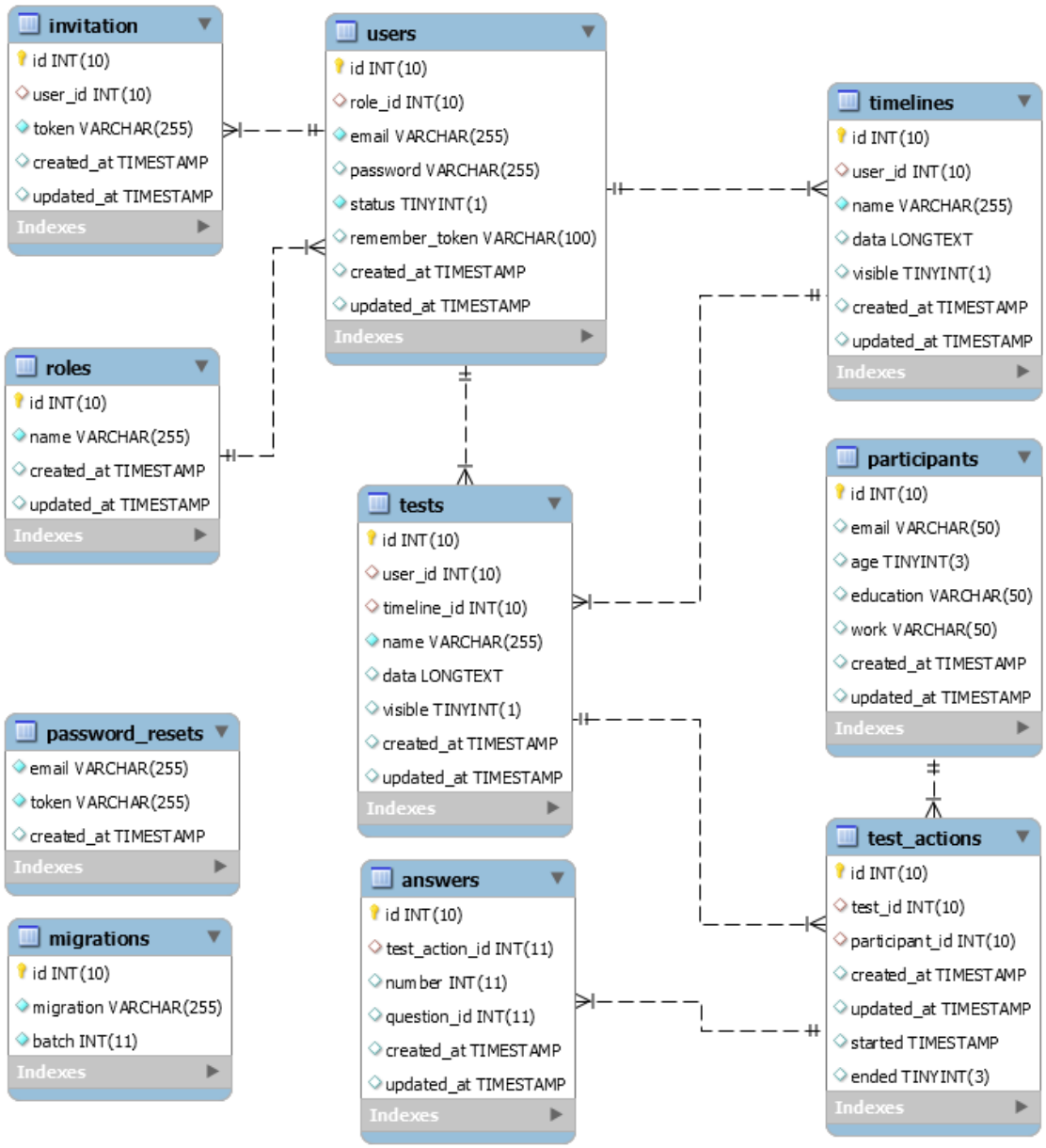
- Tvorba časové osy - V tomto módu nabízí widget pouze nástroj vyhledávání v datech a nástroj pro vytváření a editaci dat časové osy.

- Testování - V režimu testování nabízí widget nástroj pro vyhledávání a panel s testovými otázkami. Vždy se zobrazí indikátor postupu testem jako číslo aktuální otázky lomeno počet všech otázek. Spolu s ním otázka a seznam možných odpovědí. Po vyplnění může uživatel kliknout na tlačítko v pravém dolním rohu pro další otázku. Při vyplnění poslední otázky tlačítko nabízí výsledky.

- Procházení časové osy - V tomto módu widget nabízí pouze nástroj vyhledávání v datech časové osy.

Ve všech módech widgetu funguje základní funkcionalita od pana Kacerovského. Osu lze posouvat doleva a doprava šipkami na klávesnici, nebo stiskem talčítka myši a tažením. Zoom lze měnit kolečkem myši nebo klávesami + a -.

B Obrázky



Obr. 15: ERA model databáze