Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Knihovna pro tvorbu dynamických webových formulářů

Diplomová práce

Bc. Jiří Hauser

Školitel: PhDr. Miloš Prokýšek, Ph.D.

České Budějovice 2020

Bibliografické údaje

Bc. Hauser, Jiří, 2020: Knihovna pro tvorbu dynamických webových formulářů. [Library for creating dynamic web forms. Mgr. Thesis, In Czech.] – 64p., Faculty of Science, University of South Bohemia, Českéˊ Budějovice, Czech Republic.

Anotace

Tato magisterská práce se zaměřuje na problematiku definice heterogenních dat v přírodních vědách. Popisuje způsob zpracování takových dat v bio databázích a analyzuje stávající řešení dostupná online. Na základě provedených analýz navrhuje a implementuje vlastní řešení. Vytvořené řešení je publikováno jako open-source JavaScript knihovna s názvem React Form Architect, která nabízí komponenty pro uživatelské sestavení formuláře, jeho zobrazení a interpretaci stromově strukturovaných dat. Je využitelná v moderních webových prostředích a integrována do projektu univerzální nálezové databáze (UniCatDB) v rámci organizace ELIXIR.

Abstract

This master's thesis focuses on the definition of heterogeneous data in the life sciences. It is described how bio databases process such data and existing solutions available online are analyzed. Based on the performed analyzes, own solution is then designed and implemented. The created custom solution is published as an open-source JavaScript library named React Form Architect and offers components for custom assembly of the form, form display, and interpretation of tree-structured data. It can be used in modern web environments and is integrated into the universal discovery database (UniCatDB) project within the ELIXIR organization.

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 7.12.2020 Jiří Hauser

Poděkování

Děkuji panu PhDr. Miloši Prokýškovi, Ph.D. za odborné vedení mé práce, cenné rady a připomínky. Dále děkuji své rodině za poskytnutou podporu při psaní této práce.

Obsah

1 ÚVOD 1

1.1 Cíle práce 2

2 STÁVAJÍCÍ ŘEŠENÍ BIO-DATABÁZÍ 4

2.1 Rešerše 4

2.2 Analýza 4

2.2.1 GBIF 5

2.2.2 BOLDSystems 5

2.2.3 OBIS 6

2.2.4 The Australian Virtual Herbarium 6

2.3 Výsledek analýzy 6

3 ANALÝZA NÁSTROJŮ PRO SPRÁVU DYNAMICKÝCH FORMULÁŘŮ 8

3.1 Rešerše 8

3.2 Analýza 9

3.2.1 Definice hodnotících kritérií 9

3.2.2 Postup hodnocení 10

3.2.3 FormBuilder 17

3.2.4 Kinto Formbuilder 18

3.2.5 React Form Builder2 20

3.2.6 QFormBuilder 22

3.2.7 React-formio 23

3.2.8 Fl-form-builder 25

3.3 Výsledek analýzy 27

4 NÁVRH A REALIZACE VLASTNÍHO ŘEŠENÍ 34

4.1 Funkční požadavky 34
4.2 Nefunkční požadavky 35
4.3 Logický rámec 36
4.4 Scénáře použití 37
4.5 Technologie 44
4.6 Architektura RFA 44
4.7 Implementace 45
4.7.1 FormArchitect 46
4.7.2 FormRenderer 49
4.7.3 Tree 50
4.8 Dokumentace 52
4.9 Testování 53
4.10 Komparace s konkurenty 53
5 ZÁVĚR 56
6 LITERATURA 58
7 SEZNAM OBRÁZKŮ 61
8 SEZNAM GRAFŮ 62
9 SEZNAM TABULEK 63
PŘÍLOHA A – SCÉNÁŘ UŽIVATELSKÉHO TESTOVÁNÍ 64

Seznam použitých zkratek

API – Application Programming Interface

CSS – Cascading Style Sheets

ES6 – ECMAScript 2015

JSON – JavaScript Object Notation

PAAS – Platform as a service

REST – Representational state transfer

RFA – React Form Architect

UI – User Interface, uživatelské prostředí

UX – User Experience, uživatelská přívětivost

XML – Extensible Markup Language

SVG – Scalable Vector Graphics

UUID – Universally unique identifier

1 Úvod

Tato práce se zabývá problematikou definování a vizualizace dynamických webových formulářů. Konkrétněji se pak práce zaměřuje na problematiku vytváření datových schémat prostřednictvím webového uživatelského prostředí a na práci se stromově strukturovanými daty.

Přírodní vědy čelí výzvám, které přináší doba označovaná jako éra velkých dat. [1] Jednou z výzev je zpracování velkého množství dat heterogenního charakteru, které přibývají vysokou rychlostí. [2] Kolekce a analýza takových dat vede k odhalení nových poznatků, které mají pozitivní dopad na mnohá vědní, ale i nevědní odvětví. [3][4][5] Společnými znaky těchto dat je jejich obrovský objem, který je k dispozici pro zpracování, rychlost, kterou se generují nová data a jejich různorodost. Tyto společné znaky se obecně označují jako „3V" podle prvních písmen jejich anglických ekvivalentů *volume*, *velocity*, *variety,* a jsou typickými znaky pro obecnější termín *Big Data* [6].

Big Data je termín, který ovšem nemá přesně danou definici a obecně jsou takto označována data určitých znaků. Nejpopulárnějšími znaky jsou „3V" (tzv. interpretace *Gartner*) popsaná výše, která byla definovaná na začátku 21. století. Postupem času byly definované další znaky označující tato data. *IBM* ve vlastní definici „4V" rozšiřuje základní model o pravdivost dat (z angl. *veracity*) a *Microsoft* přidává navíc variabilitu a viditelnost dat ve své „6V" definici. Tento trend pokračuje a v dnešní době se setkáme s definicemi obsahujícími např. až „11V". [7]

Tato práce se zaměřuje na různorodost (*variety*, pozn. autora) velkých dat v přírodních vědách a řeší problém jejich sběru či zaznamenání, jakožto začátek jejich životního cyklu. Kolekce zahrnuje získání dat různých druhů a provedení modifikací pro jejich organizaci, zpracování a další využití [8]. V kontextu *Big Data* jsou sbíraná data heterogenního charakteru, a proto vyvstává problematika definice, zpracování a uchování dat různých datových typů a struktur.

Na straně úložišť tento problém vedl ke vzniku databází specializovaných k uchovávání různých dat bez potřeby pevně definované struktury nebo schématu – tzv. *NoSQL*[1] databází.

[1] Databázový koncept, ve kterém se pro uchování a zpracování dat nepoužívají tabulková schémata, jako je tomu u relačních databází.

Tyto databáze nabízejí, oproti tradičním relačním databázím, mimo jiné také možnost uchovávat komplexní datové struktury a efektivně reagovat na jejich strukturální změny [9]. Na opačném konci – v části aplikace, se kterou pracuje uživatel – se ovšem stále hledá řešení, které by efektivně řešilo publikování a zpracování heterogenních dat.

Standardně, jak popisuje *Microsoft Corporation* [10], *Mozilla* [11] nebo *Google LLC* [12], je úkolem programátora, aby nadefinoval pro každý typ požadovaných dat jejich strukturu a zajistil komunikaci mezi databází a částí aplikace viditelnou uživatelem. Dále zpřístupnil možnost uživateli vytvářet záznamy skrz prvky uživatelského rozhraní – u webových aplikací nejčastěji pomocí formuláře. S rostoucí sadou datových typů se zvyšují nároky na programátora, jelikož každá nová položka znamená nadefinování odpovídající datové struktury, uživatelského rozhraní a další práce s tím spojené. V oborech s vysokou mírou generování a různorodosti dat, jako jsou například přírodovědné obory, je tento přístup finančně i technicky neudržitelný.

V současné době není na trhu žádné vyhovující řešení, které by poskytovalo možnost definice datové struktury pomocí uživatelského prostředí. Respektive existující řešení jsou nevyhovující, protože nenabízejí požadovanou funkcionalitu, nejsou uživatelsky přívětivá, nejsou podporovaná nebo se nedají využít v moderních webových technologiích.

Tato práce adresuje výše zmíněný problém a hledá řešení, které by umožnilo uživateli nadefinovat vlastní datovou strukturu pomocí prvků uživatelského rozhraní, která by vedla k vytváření záznamů libovolného datového typu. Uživatel přebírá roli architekta datové struktury dále zpracovávaných informací. Tento přístup je zejména cenný v oblastech, kde je potřeba definovat a ukládat nové záznamy na denní bázi.

1.1 Cíle práce

Hlavním cílem této práce je návrh a implementace aplikace pro tvorbu dynamických formulářů na webu. Navržené řešení bude publikováno ve formě software knihovny a využito v univerzální nálezové databázi (*UniCatDB*[2]). Součástí řešení bude i systém pro práci se stromově strukturovanými daty a online dokumentace.

[2] https://www.unicatdb.org/

Tento hlavní cíl je dále rozpracován na dílčí cíle

- Analýza stávajících řešení bio-databází
- Analýza stávajících řešení nástrojů pro tvorbu dynamických webových formulářů
- Návrh a realizace knihovny

Pro dosažení cílů byly stanoveny následující úkoly

- Provést rešerši aktuálního stavu zpracování dat v bio-databázích
- Provést komparativní analýzu řešení
    - Stanovit hodnotící kritéria
    - Provést vyhodnocení
- Vytvořit vlastní řešení
    - Definovat funkční a nefunkční požadavky
    - Vytvořit návrh architektury knihovny
    - Implementace
    - Testování
    - Dokumentace

2 Stávající řešení bio-databází

V této kapitole je provedena rešerše a analýza stávajících řešení bio-databází dostupných online se zaměřením na jejich přístup ke kolekci různorodých dat.

V posledních deseti letech dochází k vysokému nárůstu počtu záznamů online bio-databází. Příkladem může být například databáze *The Global Biodiversity Information Facility*, která zaznamenala pětinásobný nárůst uchovávaných dat a pokořila milník 1 miliardy uchovávaných záznamů [13]. I přes rostoucí trend digitalizace bio-databází se odhaduje, že pouze 10 % všech biologických kolekcí je dostupných v digitální podobě [14]. Z toho důvodu je potřeba zkoumat možnosti kolekce, identifikovat slabá místa a ty se poté pokusit vylepšit.

2.1 Rešerše

Rešerše vychází z rozsáhlé studie [14], která kvantitativně zkoumala využívání bio-databází ve vědeckých publikacích od roku 2010 do roku 2017 a trendy ve využívání biologických dat. Analyzováno bylo přes 1900 publikací, z nichž bylo na základě daných kritérií vybráno přes 500 nejrelevantnějších. V rámci studie vznikla tabulka deseti nejcitovanějších bio-databází, z kterých autor vybral pro analýzu ty, které byly citované alespoň 20krát. Seznam vybraných databází je zobrazen v tabulce 1.

Tabulka 1 Nejcitovanější bio-databáze

Databáze	**Počet citací**
GBIF	155
BOLDSystems	27
OBIS	21
The Australian Virtual Herbarium	20

2.2 Analýza

Tato podkapitola analyzuje způsob zpracování nahrávaných dat při jejich publikaci v jednotlivých bio-databázích. Výsledek analýzy je popsán v podkapitole 2.3.

2.2.1 GBIF

The Global Biodiversity Information Facility (dále *GBIF*) je mezinárodní biologická databáze se sídlem v dánské Kodani. Za pomoci účastnických uzlů poskytuje po celém světě infrastrukturu pro správu a definici biologických dat. *GBIF* uchovává enormní množství dat vztahujících se k biodiverzitě. [13]

GBIF je veřejně dostupná databáze skrze webový portál a nabízí vyhledávání a filtrování biologických druhů včetně jejich fotografií, mapy výskytu, taxonomických stromů a agregovaných metrik.

GBIF definuje komplexní postup publikování dat uživatelům, kteří musí být pod záštitou spřátelené organizace. Dovoluje publikovat datové soubory prostřednictvím svého proprietárního open-source softwarového nástroje s názvem *IPT* (*Integrated Publishing Toolkit*). Nahrávaná data musí projít několika kroky transformací před samotným zveřejněním. Jedním krokem je mapování dat podle standardu *Darwin Core*[3]. Data, která se nepodaří namapovat na statickou datovou strukturu popsanou standardem jsou vyfiltrována a zahozena.

GBIF také umožňuje správu dat programově prostřednictvím *REST API*. Uživatel ovšem nemůže pomocí tohoto *API* definovat a nahrávat do systému nová data.

2.2.2 BOLDSystems

The Barcode Of Life Data System (dále *BOLD*) je databáze uchovávající tzv. „Čárové kódy DNA“[4] se sídlem v Guelphu v kanadském Ontariu. V rámci své webové platformy nabízí portál pro vyhledávání dat, edukační portál, databázi *BIN* (*Barcode index number*) údajů a platformu pro správu dat. [15]

BOLD poskytuje na svých stránkách návod k využívání nabízené platformy, která obsahuje i postup přispívání do databáze. Přihlášení uživatelé mohou nahrát data pomocí tabulkového souboru, který *BOLD* předkládá uživateli ve formě šablony, a specifikuje množinu polí, které musí být povinně vyplněny, aby data prošla validací a mohla být úspěšně zpracována. Další

[3] Darwin Core je standardizovaný slovník identifikátorů sloužících pro popis biologických dat.

[4] Čárový kód života je krátká část genu sloužící k identifikaci druhu organismu.

možností je využití online formuláře, jehož pomocí uživatel vyplní informace pro jednotlivé datové záznamy. Nabízené formulářové prvky kopírují strukturu šablony tabulkového souboru.

BOLD umožňuje sbírat data pomocí veřejného *API*, ale stejně jako *GBIF* neumožňuje nahrávání dat jakýmkoliv způsobem.

2.2.3 OBIS

Ocean Biodiversity Information System (dále *OBIS*) je databáze zaměřená na biodiverzitu podmořského světa. Je složena z účastnických uzlů po celém světě se sídlem v belgickém Oostende. Nabízí online platformu pro vyhledávání a publikování dat. [16]

OBIS, jak uvádí na svém webu [16], úzce spolupracuje s bio-databází *GBIF*. Pravidla pro nahrávání vlastních dat jsou proto totožná s těmi, které na svých stránkách specifikuje *GBIF*, a která jsou popsaná v podkapitole 2.2.1. Pro přispívání do databáze se využívá stejný software jako v případě *GBIF*, a data se tedy i stejně mapují na standardizovanou statickou strukturu.

OBIS nabízí veřejné *REST API*, ale stejně jako v případě *BOLD* toto *API* slouží jen pro získávání dat a není možné jeho pomocí data nahrávat.

2.2.4 The Australian Virtual Herbarium

The Australian Virtual Herbarium (dále *AVH*) je databáze zaměřená na rostlinnou biodiverzitu oblasti Austrálie a Nového Zélandu. Je spravována *CHACH* (*Council of Heads of Australasian Herbaria*) se sídlem v australském městě Canberra. [17]

AVH na svých stránkách nijak nepopisuje způsob přispívání do databáze. Dočteme se však, že jsou postupně přidávány a aktualizovány nové záznamy dle herbářů spřátelených organizací, jejichž seznam následuje na webu v textu [17]. Lze se tedy domnívat, že nahrání dat probíhá interně.

2.3 Výsledek analýzy

Bio-databáze v dnešní době nabízejí veřejně dostupné webové platformy pro vyhledávání a správu dat. Pokročilejší řešení nabízí možnost uživatelům nahrávat vlastní data a přispět tak svým dílem komunitě. Za nejpokročilejší příklad v tomto ohledu můžeme považovat *GBIF*, která nabízí nejrobustnější platformu z hlediska funkcionality, a jejíž volně dostupný software pro publikování dat využívají i ostatní bio-databáze.

Ve všech zkoumaných případech musí nahrávaná data svojí strukturou odpovídat specifikovanému schématu. Proto jsou zkoumané databáze zaměřené na určitý typ uchovávaných dat. Při potřebě uchovávat záznamy, které svojí strukturou neodpovídají danému schématu, jsme nuceni lišící se část informací zahodit nebo vytvořit novou databázi, která bude přijímat data strukturovaná dle našich potřeb. Tím ale přijdeme o potencionální vazby a souvislosti, které by mohly ve spojení s dalšími záznamy vzniknout. Právě prostředí s daty, která jsou propojená, představují vysoký potenciál pro získávání nových poznatků [18]. Abychom mohli takové prostředí vytvořit, musíme nejprve vyřešit způsob sběru a uchování různorodých dat. Logickým řešením se zdá být formulář, jehož obsah je dynamický dle potřeb nahrávaných dat.

3 Analýza nástrojů pro správu dynamických formulářů

V této kapitole je provedena rešerše a analýza stávajících nástrojů pro tvorbu a vykreslení dynamických formulářů v jazyce *JavaScript*. Rešerše je provedena na základě vymezení klíčových slov a rešeršního dotazu, jak je blíže pospáno v podkapitole 3.1. Analýza je podrobně popsána v podkapitole 3.2. Výsledky analýzy jsou shrnuty na konci kapitoly.

3.1 Rešerše

Rešerše byla provedena na základě vyhledávání pomocí rešeršního dotazu. Jako informační zdroj byl použit vyhledávací portál *Google.com*, který poskytuje rozšířenou funkcionalitu vyhledávání pomocí operátorů. Pomocí nich byl omezen rozsah vyhledávání na server *Github.com,* jakožto na největší *webhosting* pro open-source projekty na světě [19].

Výsledný rešeršní dotaz má tvar „*site:github.com (form OR builder OR formbuilder OR dynamic form OR react form builder) AND javascript"*. Souhrnně jde o vyhledávání obsahu na serveru *Github.com*, který obsahuje alespoň jedno z nepovinných klíčových slov a zároveň všechny povinná klíčová slova. Jako nepovinná jsou určena slova: *form, builder, formbuilder, dynamic form, react form builder* a jako povinné je určeno slovo *javascript.*

Do výsledného analyzovaného vzorku byly zahrnuty jen projekty, které jsou publikovány v *registru npm*[5], protože *Github.com* obsahuje i projekty, které jsou ve fázi vývoje nebo byl jejich vývoj ukončen před zveřejněním. Dále byly výsledky omezeny na projekty, které nabízejí možnost definice formuláře pomocí uživatelských prvků a nejsou starší tří let – bráno z pohledu poslední aktivity v repositáři.

Z výsledného vzorku musely být odebrány knihovny *zubair-react-form-builder* a *react-form-builder*, které se nepodařilo nainstalovat a spustit z důvodu interní chyby, kterou obsahují. Výsledky rešerše jsou zobrazeny v tabulce 2.

[5] Registr npm je veřejná sbírka balíčků open-source kódu pro Node.js, front-end webové aplikace, mobilní aplikace, roboty, směrovače a nespočet dalších potřeb JavaScript komunity.

Tabulka 2 Výsledky rešerše nástrojů pro tvorbu dynamických formulářů

Název	**Url**	**Počet hvězdiček**	**Poslední příspěvek**
fl-form-builder	https://github.com/fourlabsldn/fl-form-builder	12	23.05.2018
react-form-builder2	https://github.com/Kiho/react-form-builder	131	12.12.2019
formBuilder	https://github.com/kevinchappell/formBuilder	1800	02.02.2020
QFormBuilder	https://github.com/baggachipz/q-form-builder	42	04.10.2019
Formbuilder	https://github.com/Kinto/formbuilder	538	15.03.2019
react-formio	https://github.com/blackjk3/react-form-builder	92	17.03.2020

3.2 Analýza

Analýza je provedena na základě hodnotících kritérií, která jsou definována v oddílu 3.2.1. Oddíl 3.2.2 vysvětluje postup hodnocení a obsahuje podrobný popis jednotlivých kritérií. Dále jsou popsaná jednotlivá analyzovaná řešení včetně jejich silných a slabých stránek. Závěrem podkapitoly je shrnutí a výsledek analýzy.

3.2.1 Definice hodnotících kritérií

Proces definice kritérií pro hodnocení *JavaScript* knihoven specializujících se na tvorbu webových dynamických formulářů vychází z více informačních zdrojů. Do výsledné množiny kritérií byly zahrnuty rady a postupy pro obecné porovnávání softwarových knihoven z článků [20] a [21]. K výběru dále přispěla autorova zkušenost s vývojem webových aplikací a diskuse se seniorními vývojáři z oboru. Vzhledem ke specifické povaze porovnávaných řešení, kdy je potřeba brát v potaz i uživatelskou přívětivost aplikace, je věnována pozornost hodnotícím kritériím ze strany vývojáře i ze strany uživatele. Vznikly tak dvě hodnotící sady kritérií – *uživatelská* a *vývojářská*. Každá ze sad obsahuje specifická hodnotící kritéria a každé kritérium je hodnoceno na určené škále. Tento způsob hodnocení zajistil, že hodnocení bylo co nejobjektivnější vzhledem ke kvalitativní povaze daného úkolu.

Vývojářská kritéria jsou rozdělena do hodnotících sekcí. Sekce jsou sestaveny jako dichotomické uzavřené otázky, a proto při hodnocení obdrží jednotlivá sekce maximální, nebo naopak minimální ohodnocení. Protože existují případy, kdy by tento typ uzavřené otázky byl příliš omezující, jsou některé sekce hodnoceny na základě měřeného faktoru, jako např. počet přispívajících členů do kódu. V takovém případě je určena hodnota nejlepšího výsledku měřené knihovny jako referenční, hodnocena maximem bodů. Zbylé výsledky jsou hodnoceny poměrově vůči referenční hodnotě. V tabulce 3 jsou vypsaná jednotlivá *vývojářská* kritéria, jejich popis a vysvětlení způsobu hodnocení.

Uživatelská kritéria jsou zvolena na základě doporučených postupů *UX* testování z publikací [22], [23] a článku [24]. Jsou rozdělena na *subjektivní* a *objektivní*. Subjektivní kritéria, jako např. *vzhled aplikace*, jsou hodnocena na škále 1 – *nejlepší* až 5 – *nejhorší.* Objektivní kritéria, jako např. *množství času potřebného k dokončení úkolu,* jsou hodnoceny jejich naměřenou hodnotu, která byla zjištěna experimentálně. Popis jednotlivých *uživatelských* kritérií je uveden v tabulce 4. Metoda uživatelského testování vybraných nástrojů je popsána v oddílu 3.2.2.

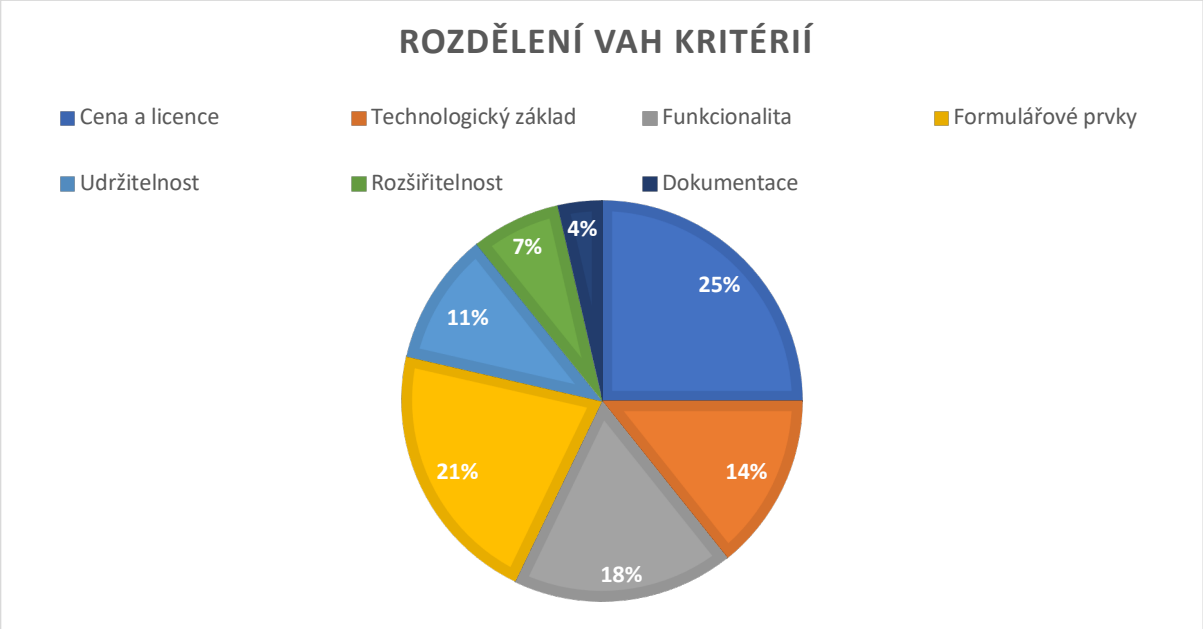
3.2.2 Postup hodnocení

Z vývojářského hlediska byly analyzované knihovny hodnoceny autorem práce. Každá knihovna byla analyzována ve svém repositáři na serveru *github.com*, kde byla zkoumána hodnotící kritéria *V1*, *V5* a *V7*. Knihovna byla poté naklonována, nainstalována a lokálně spuštěna podle uvedeného postupu v dokumentaci, nebo improvizovaně, pokud takový postup nebyl dostupný. Autor práce podrobil knihovny jednotnému úkolu – vytvořit objednávkový formulář pro návštěvníky smyšleného e-shopu v co nejrozsáhlejší možné verzi, tedy s využitím co nejvíce různých formulářových prvků a jejich editací a validací. Při plnění úkolu autor zkoumal kritéria *V3* a *V4*. Nakonec autor procházel zdrojový kód, závislosti a použité technologie každé knihovny. Tím ohodnotil zbylá kritéria *V2* a *V6*.

Hodnocení vývojářských kritérií probíhalo podle *vícekriteriální analýzy variant.* Konkrétněji podle metody *TOPSIS*, která je založena na eukleidovských vzdálenostech. Tato metoda určuje *kompromisní řešení* podle nejkratší vzdálenosti od *ideálního řešení*, resp. nejdelší vzdálenosti od *bazálního řešení*. Jako *ideální řešení* je určena hypotetická varianta, tedy knihovna, která je ve všech kritériích hodnocena nejlepšími možnými hodnotami. *Bazální řešení* je opakem ideální varianty, tedy teoreticky nejhorší knihovna hodnocena úplným minimem ve všech

kritériích. Určené kompromisní řešení si můžeme představit jako knihovnu, která se nejvíce blíží ideálnímu řešení. [25]

Vývojářským kritériím byla určena váha podle *metody pořadí.* Tato metoda umožňuje přiřadit normalizované váhy kritériím podle jejich důležitosti. Autor subjektivně seřadil kritéria dle důležitosti a pomocí *metody pořadí* vypočítal jejich váhy. U sekcí jednotlivých kritérií bylo postupováno stejným způsobem. Výsledné rozložení vah je zobrazeno na grafu Graf 1. Konkrétní kritéria jsou zobrazena v tabulce 3.



Graf 1 Rozdělení vah vývojářských kritérií

Tabulka 3 Vývojářská hodnotící kritéria

Symbol	Kritérium
V1	**Cena a Licence** Cena a licence řešení je zpravidla první a klíčový faktor, na který se hodnotitelé zaměřují při výběru software. Proto toto kritérium hodnotí náklady, které vzniknou při využití, vývoji a publikování vybrané knihovny. Jsou zde započítány i náklady na podporu využívaného produktu. Přiřazená hodnota tomuto kritériu udává cenu v dolarech. Pokud je produkt dostupný v různě drahých variantách, je zvolena nejlevnější varianta.
V2	**Technologický základ** Zahrnutím knihovny do projektu přijímá vývojový tým část cizího kódu do své vlastní aplikace. Z toho důvodu je důležité zhodnotit využité technologie a závislosti, které mohou mít negativní dopad na fungování aplikace. Jelikož se hodnotí nástroj využitelný v moderním webovém prostředí, měl by být spustitelný v prostředí založeném na *React.js.* Nejvyšší možné ohodnocení dostane takové řešení, které bude poskytovat typovou definici nebo bude napsané v jazyce *TypeScript*[6] s uživatelským rozhraním v *React.js* ve verzi *16.8.0*[7] a vyšší. Nejhorší skóre naopak dostane řešení, které nelze kvůli použité technologii využít.
V3	**Funkcionalita** Funkcionalita knihovny na tvoření dynamických webových formulářů by měla poskytovat možnost uživatelského exportu a importu schématu ve formě *JSON* dokumentu. Knihovna by měla být schopna vykreslení formuláře na základě schématu a vložení i získání hodnot formuláře. Vývojář by měl být schopen základního přizpůsobení vzhledu na úrovni přebarvení prvků. Tvořený formulářový prvek by měl být duplikovatelný, smazatelný a upravitelný na úrovni atributů. Uživateli by měla být zpřístupněna možnost nastavení základní

[6] *Typescript* je typová nadstavba *JavaScript* sloužící pro dřívější detekci chyb a psaní kvalitnějšího kódu.

[7] *React.js* verze *16.8.0* přinesla nové API a sadu inovací.

	i rozšířené validace jednotlivých prvků. Knihovna by měla umožnit shlukování prvků do skupin. Nástroj disponující výše zmíněnou funkcionalitou obdrží nejvyšší ohodnocení.
V4	**Formulářové prvky** Z hlediska nabízených formulářových prvků pro tvorbu formuláře by knihovna měla podporovat nejlépe všechny tyto formulářové prvky: • Button • Checkbox • Checkbox Multiple • Date • File upload • Header • Hidden input • Number • Paragraph • Radio • Range • Rating • Select • Select Multiple • Text field • Text area Knihovna, která nabídne uživateli všechny nabízené formulářové prvky, obdrží nejvyšší hodnocení. Pokud knihovna naopak nepodporuje žádný z prvků, dostane hodnocení nejhorší.

V5	**Udržitelnost** Cílem vývojářského týmu je dlouhodobá dostupnost aplikace a všech jejích částí. Tento faktor je důležitým ukazatelem kvality, proto se od hodnoceného nástroje očekává dostupnost v dlouhodobém horizontu. Indikátory dlouhodobé proveditelnosti jsou popularita, četnost aktualizací a aktivní vývojový tým. Popularita je měřena na základě kladných reakcí vyjádřených počtem hvězdiček na portálu g*ithub.com.* Faktorem správného chování aktualizací jsou krátké aktualizační cykly a rychlé opravování chyb. Nejvyšší skóre získá knihovna s vysokou pravděpodobností dlouhodobého rozvoje. Naopak nástroj, jehož vývoj byl ukončen nebo delší dobu stagnuje, získá skóre nejnižší.
V6	**Rozšiřitelnost** Z pohledu vývoje je důležité, aby byl nástroj rozšiřitelný a upravitelný pro specifické potřeby systému, do kterého je implementován. Nástroj by měl být pokryt jednotkovými a end-to-end testy pro automatickou kontrolu správnosti a funkcionality kódu. Nejlépe hodnocen bude takový nástroj, který zpřístupní *API* pro přidání vlastních formulářových prvků a bude pokryt aktuálními výše zmíněními testy. Nejhorší ohodnocení dostane takový nástroj, který není otestován, neposkytuje žádnou možnost rozšíření a zároveň není vyvíjen jako open-source.
V7	**Dokumentace** Softwarová knihovna není příliš užitečná, pokud se s ní nenaučí vývojáři pracovat. Proto je důležitou součástí softwarového balíčku i dokumentace, jak tvrdí např. Brookshear [28]. Dokumentace pomáhá vývojářům v učení a při implementaci knihovny. Měla by poskytovat popis *API* knihovny, návod na instalaci a příklady užití v základní a více detailní verzi. Měla by být přehledná a strukturovaná. Nejvyšší skóre dosáhne taková knihovna, jejíž dokumentace bude strukturovaná, bude obsahovat návod na instalaci, základní i pokročilé příklady užití a popis *API*.

Pro ohodnocení knihoven z uživatelského hlediska byla zvolena *UX* metoda uživatelského testování *Comparative Usability Testing,* která umožňuje porovnat software z uživatelského hlediska s jeho konkurenty. Podle této metody vybraný reprezentativní vzorek uživatelů plní zadaný úkol v každém z testovaných nástrojů, přičemž je pozorováno jejich počínání a jsou měřeny předem určené faktory. Metoda zároveň radí, jak postupovat při vybírání účastníků testování, komunikaci s nimi a zpracování naměřených výsledků. Počet účastníků testování je variabilní a určován autorem testování. Podle studií [26] a [27] stačí již 5 účastníků k odhalení 80 % potencionálních problémů použitelnosti testované aplikace.

Účastníci testování byl vybráni na základě očekávané množiny budoucích koncových uživatelů vyvíjené knihovny. Dle autorova názoru do této množiny patří vzhledem k orientaci na akademické prostředí obecně technicky zdatnější lidé, s dobrou zkušeností s webovým prostředím a základní orientací v informačních a komunikačních technologiích. Vybráno bylo 6 účastníků testování, kteří splňují výše uvedenou specifikaci, z toho 2 ženy a 4 muži. Jeden z účastníků byl student doktorského studijního programu s orientací na ekonomii, tři studenti vysoké školy se zaměřením na IT, jeden pracovník softwarového vývoje se zaměřením na UX a zaměstnanec státní správy s vysokoškolským titulem. Každému účastníkovi byly zajištěny stejné podmínky pro testování.

Testování knihoven probíhalo podle sestaveného scénáře (Příloha A) a účastnil se ho vždy jen jeden účastník. Autor se v úvodu představil a vysvětlil důvody, způsob a průběh testování. Poté byl účastníkovi zadán úkol, který měl splnit s pomocí každé z testovaných knihoven. Úkol byl stejný pro všechny účastníky testování. Byl zvolen tak, aby byl splnitelný ve všech testovaných nástrojích, byl jednoduchý a přinutil účastníka využít co nejvíce různých formulářových prvků. Během plnění úkolu autor měřil čas a zaznamenával počet chyb, které účastník udělal. Tím zkoumal *objektivní* hodnotící kritéria *U5* a *U6*. Po splnění úkolu vyplnil uživatel dotazník, který obsahoval *subjektivní* hodnotící kritéria *U1*, *U2*, *U3*, *U4*. Tento postup se opakoval, dokud nebyly otestovány zbylé knihovny.

Pořadí testování knihoven se po každém uživateli změnilo, protože testované knihovny jsou si funkcionálně podobné. Při zachovaní stejného pořadí pro všechny účastníky by byly první testované knihovny znevýhodněny, protože by se na nich účastníci naučili postupy tvoření formuláře, které by poté využili při testování následujících nástrojů. Konkrétní uživatelská hodnotící kritéria jsou zobrazena v tabulce Tabulka 4.

Tabulka 4 Uživatelská hodnotící kritéria

Symbol	Kritérium	Typ
U1	**Vzhled aplikace** Vyladěný vzhled, důraz na detail a celistvost dizajnu je dnes považováno za standard a očekáváno od moderní aplikace. V důsledku nedodržení těchto faktorů působí aplikace amatérským dojmem a může odradit potencionálního uživatele od jejího používání. [23] Účastníci testování hodnotí vzhled aplikace podle osobních preferencí na škále 1 až 5.	Subjektivní
U2	**Plynulost aplikace** Toto kritérium hodnotí prodlevy aplikace při jejím průchodu. Účastníci testování hodnotí plynulost aplikace podle osobního dojmu při plnění zadaného úkolu na škále 1 až 5.	Subjektivní
U3	**Pocit z aplikace** Účastníci testování hodnotí celkový pocit z aplikace, který měli po dokončení zadaného úkolu. Hodnocení na stupnici 1 až 5 vyjadřuje jak silný pozitivní, nebo naopak negativní pocit zanechala aplikace na uživateli po jejím použití.	Subjektivní
U4	**Obtížnost** Účastníci testování hodnotí, jak obtížné bylo dokončit zadaný úkol za použití testované knihovny. Nejlépe hodnocen by měl být nástroj, který se zdál uživateli nápomocný při plnění úkolu a tím snížil obtížnost jeho dokončení na minimum. Nejhorší hodnocení dostane takový nástroj, který co nejvíce ztíží nebo dokonce znemožní uživateli dokončit zadaný úkol.	Subjektivní

U5	**Množství času potřebného k dokončení úkolu** Kritérium udává množství času měřené v minutách, které je potřebné k dokončení zadaného úkolu účastníkem testování.	Objektivní
U6	**Počet chyb** Kritérium udává počet chyb, které uživatel při plnění úkolu udělal. Za chybu se považuje akce vedoucí k nesprávnému dokončení úkolu. Příkladem je použití chybného formulářového prvku, které značí problém v jeho pojmenování.	Objektivní

3.2.3 FormBuilder

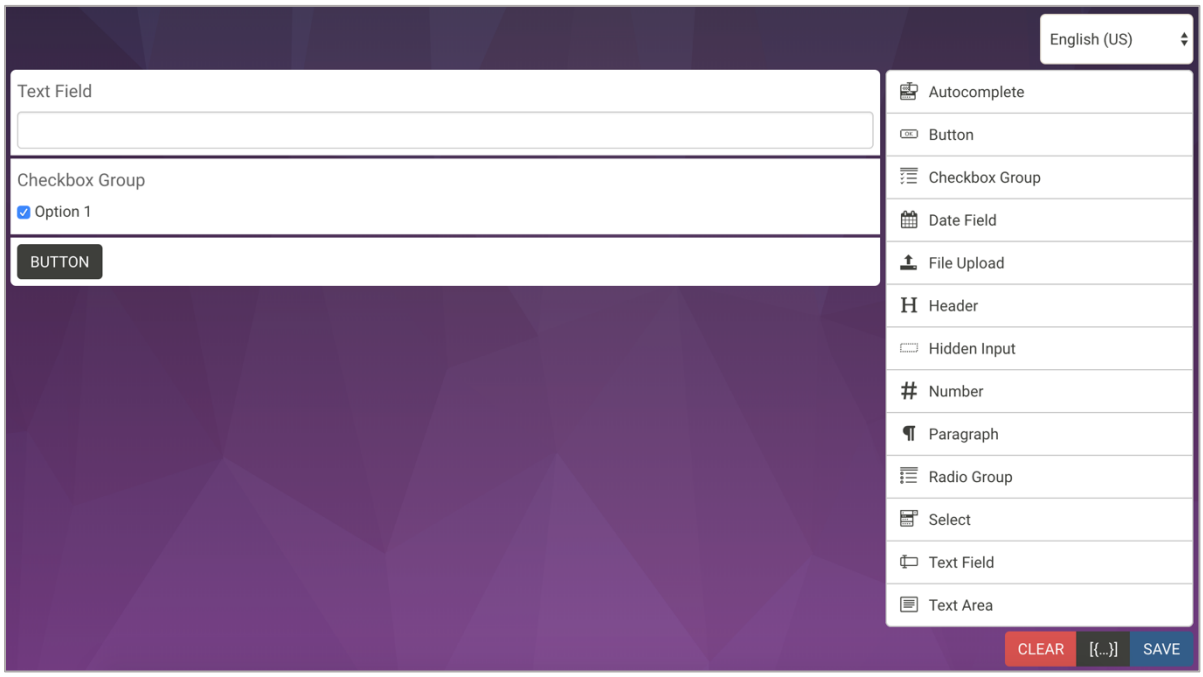
Lídrem z hlediska oblíbenosti v oblasti tvoření dynamických formulářů je *jQuery* plugin *FormBuilder*, který obsahuje i separátní plugin pro vykreslování formuláře *FormRender*. Nástroj je k dispozici zdarma pod licencí *MIT* a je vyvíjen jako open-source. Aktuálně je ve verzi 3.4.0 a disponuje aktivní komunitou uživatelů. V době psaní této práce je hodnocen téměř 1800 hvězdičkami a má 135 otevřených a 596 uzavřených nálezů. Nástroj je napsaný v jazyce *JavaScript*.

FormBuilder nabízí uživatelsky přívětivé prostředí *drag and drop* a 13 prvků pro stavbu formuláře. Každý prvek formuláře nabízí možnost smazání, duplikace, upravení atributů a nastavení *CSS* třídy. *FormBuilder* generuje schéma ve formě *JSON* nebo *XML*, podle kterého poté *FormRender* vykreslí formulář. Dokumentace knihovny je rozsáhlá a obsahuje příklady užití knihovny v různých prostředích.

Výhodou je možnost lokalizovat plugin do jiného jazyka, uživatelské rozhraní, oddělení částí pro tvorbu a vykreslení formuláře, interaktivní dokumentace a možnost rozšíření funkcionality. Knihovna obsahuje možnost základní *HTML5* validace volitelnosti jednotlivých prvků, tedy vyzve uživatele k vyplnění pole, pokud je označeno jako povinné a zároveň není vyplněné.

Nevýhody nástroje vyplývají hlavně z použití technologie *jQuery*. Pro integraci do aplikace vyvíjené v *React.js* je potřeba import celé knihovny *jQuery* a nastavení globálních proměnných objektu *window* pro zpřístupnění *jQuery* operátorů. Dále přecházení ze standardu ES6 na

dřívější verze pro import a práci s knihovnou. Knihovně *FormBuilder* navíc chybí automatizované testování. Z hlediska nabízených funkcí chybí rozšířená validace dat. Knihovna neposkytuje možnost shlukování formulářových prvků do skupin bez použití externího nástroje.



Obrázek 1 FormBuilder

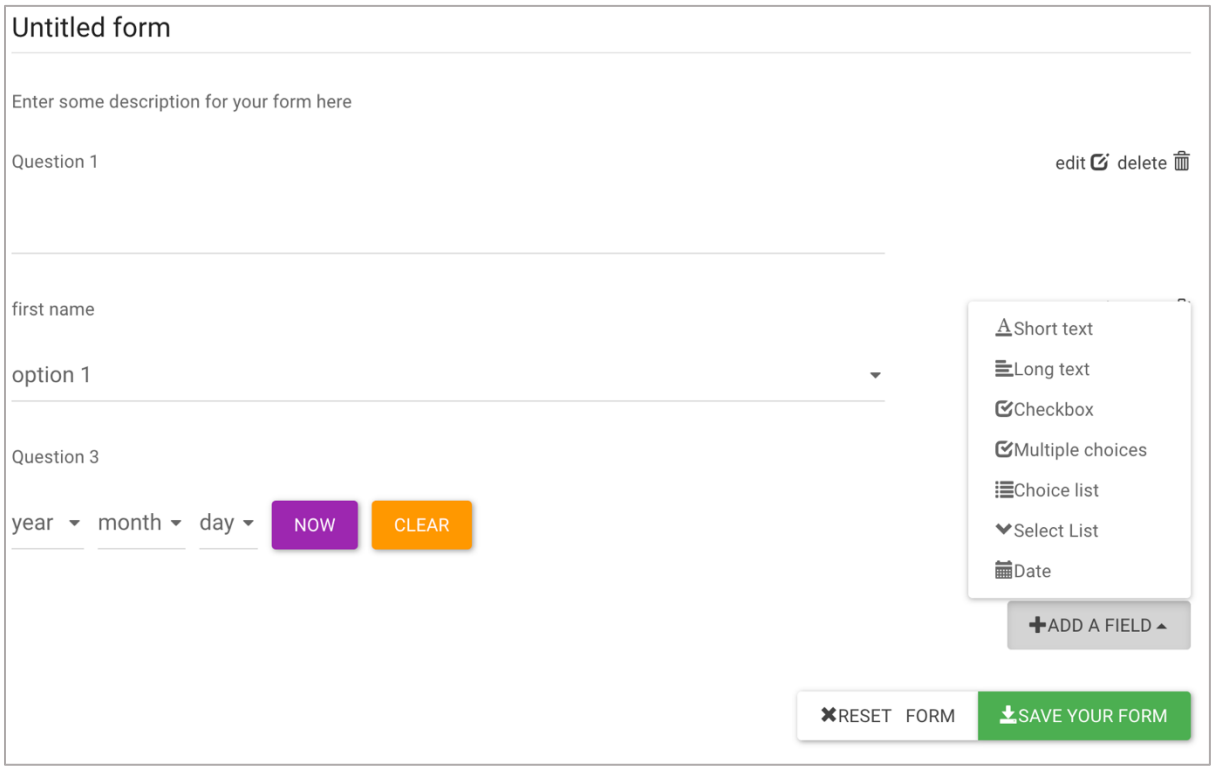
FormBuilder nabízí zdařilé řešení pro tvorbu a zobrazení dynamických formulářů, které ocení zejména programátoři a správci systémů založených na *jQuery*. Největší využití knihovna najde v redakčních systémech jako je například *Wordpress*[8], kde v administraci uživatel pomocí pluginu *FormBuilder* vytvoří jednoduchý formulář a na produkci se plugin *FormRender* postará o jeho vykreslení. Použití nástroje v moderní webové aplikaci založené na *React.js* bychom se měli spíše vyhnout, protože knihovna zvýší nároky celé aplikace kvůli importu a práci s knihovnou *jQuery*. Absence testů zvyšuje riziko neočekávaných chyb.

3.2.4 Kinto Formbuilder

Kinto je open source minimalistická služba pro ukládání *JSON* dat a jejich sdílení a synchronizaci [29]. Ve své nabídce přichází i s řešením pro generování dynamických formulářů s názvem *Formbuilder* (pro přehlednost bude dále označováno jako *Kinto*

[8] https://wordpress.org/

Formbuilder). Knihovna je nabízena zdarma pod licencí *Apache-2.0* jako open-source, je ohodnocena 538 hvězdičkami a má 58 otevřených a 69 uzavřených nálezů. *Kinto Formbuilder* využívá technologický základ *JavaScript* a *React.js*. Vývoj knihovny se zastavil v březnu roku 2019 a repositář je nyní archivován.



Obrázek 2 Kinto FormBuilder

Nástroj nabízí uživatelské rozhraní pro tvorbu dynamických formulářů a 7 formulářových prvků. Data exportuje ve formě *JSON* schématu na server *Kinto* a pro vykreslení formuláře využívá knihovnu *react-jsonschema-form*[9]. Prvky formuláře nabízejí možnost úpravy atributů a mazání prvků. Dokumentace je stručná a nabízí jen základní popis funkcionality. Knihovna obsahuje neaktuální jednotkové testy.

Výhody a nevýhody značně závisí na způsobu využití knihovny. Hlavním záměrem je rychlé vytvoření formuláře – dotazníku a sběr získaných dat ze serveru. Tomu odpovídá i omezené a specificky zaměřené pole dostupných formulářových prvků a limitovaná možnost úpravy

[9] Open source React.js knihovna pro vykreslení formuláře na základě JSON schématu. (02.2020, dostupné z: https://github.com/rjsf-team/react-jsonschema-form)

jejich atributů. Tento účel plní *Kinto Formbuilder* dobře a je pochopitelná i integrace s *Kinto* serverem.

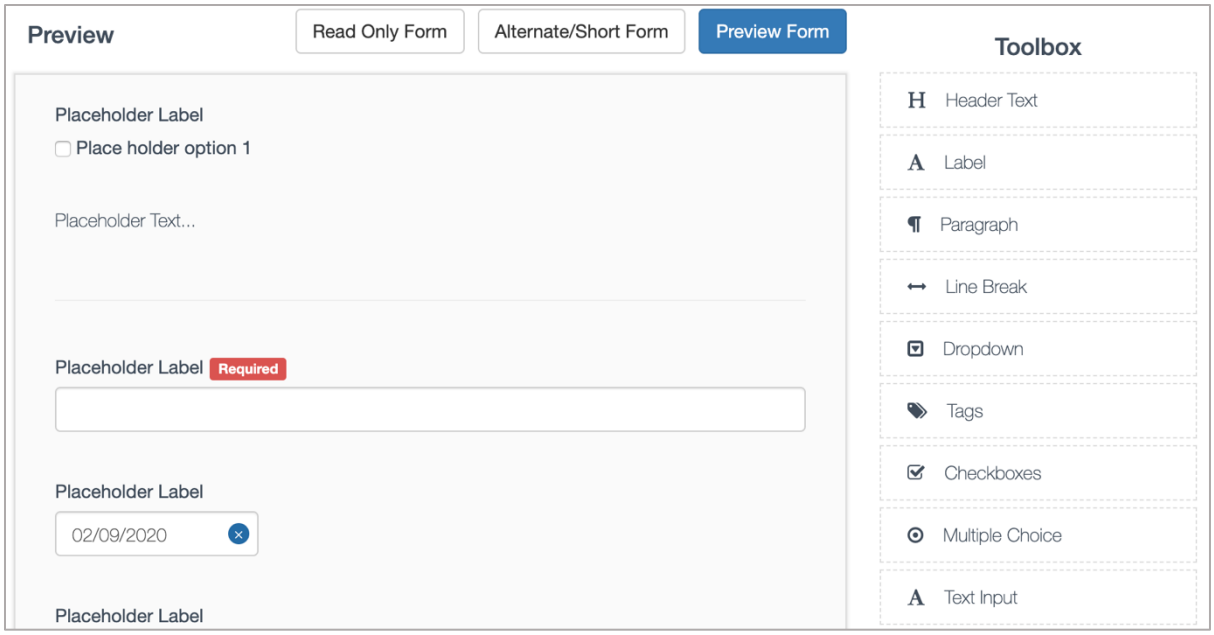
Zcela chybí možnost nastavení validace nebo shlukování prvků do skupin. Uživatelské rozhraní je nedotažené a nese známky grafické nesourodosti. Slabinou je také chybějící důraz na *UX*. Třídění prvků formuláře lze provádět *drag and drop* způsobem, ovšem chybí signalizace a odezva prvků na prováděné akce. Nevýhodou je také omezená škála dostupných formulářových prvků a chybějící *API* pro její rozšíření.

Kinto Formbuilder je zajímavý projekt s úzkým záběrem využití. Jeho vývoj je v současné době ukončen. Z tohoto důvodu a výše zmíněným negativům se knihovna nehodí pro obecné využití.

3.2.5 React Form Builder2

React form builder 2 je jedna z mnoha knihoven vycházejících z projektu *React form builder*[10]. Se svými 130 hvězdičkami se staví mezi jeho nejúspěšnější následovatele. Knihovna je k dispozici zdarma pod licencí *MIT* a je vyvíjena jako open-source. Aktuálně je ve verzi 0.4.3, je dále aktualizována a má 3 otevřená a 47 uzavřených nálezů. Knihovna je napsána v jazyce *JavaScript* a využívá *React.js*.

[10] Open source knihovna pro tvoření dynamických formulářů v React.js. Vývoj byl ukončen v roce 2017 (dostupné z https://github.com/blackjk3/react-form-builder)



Obrázek 3 React Form Builder

Nástroj nabízí komponenty pro uživatelskou tvorbu formuláře *ReactFormBuilder* a pro jeho vykreslení *ReactFormGenerator*. Komunikaci mezi komponentami zajišťuje *JSON* web *API endpoint*, který musí vývojář specifikovat, popř. nadefinovat. *ReactFormBuilder* obsahuje uživatelsky přívětivé prostředí *drag and drop* pro tvorbu formuláře. Při sestavování má uživatel k dispozici 16 formulářových prvků. Každý prvek lze upravit na úrovni atributů nebo smazat. Dokumentace vychází z původní verze knihovny *React form builder* a obsahuje již neaktuální informace, které rozšiřuje o vlastní příklady užití.

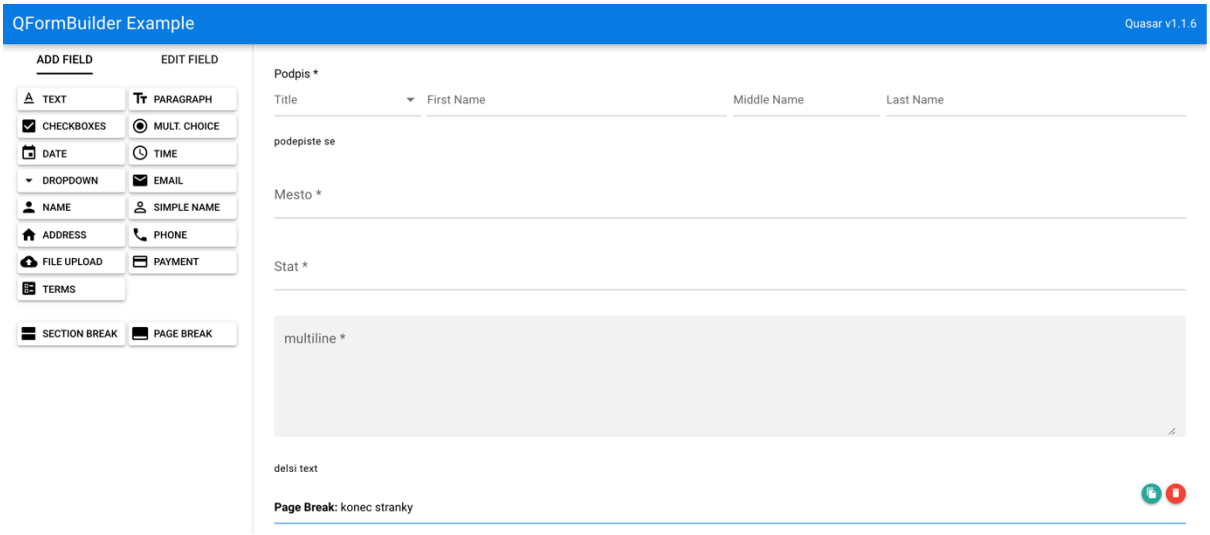
Výhodou nástroje je uživatelsky přehledné prostředí, široká škála formulářových prvků a možnost náhledu formuláře při jeho tvorbě. Nástroj nabízí i specializované prvky jako je např. *Signature,* tedy pole sloužící pro podpis. Další výhodou je i aktivní snaha autora řešit nálezy a knihovnu rozvíjet.

Nevýhodou knihovny je potřeba *web API endpointu* pro komunikaci mezi komponentami pro tvoření a vykreslení formuláře. Integrace nástroje závisí na porozumění zdrojového kódu, neboť dokumentace obsahuje aktuální i zastaralé, již nefungující popisy a odkazy. Navíc chybí popis *API* knihovny a základní ukázky použití, na kterých by vývojář mohl dále stavět. Nástroj není automatizovaně testovaný. Z hlediska funkcionality chybí možnost validace prvku, jeho duplikace v rámci formuláře a možnost shlukování prvků do skupin. Nástroj nelze graficky přizpůsobit podle systému do kterého je integrován.

React form builder 2 nabízí komplexní řešení pro tvorbu dynamických formulářů, které využívá *React.js* pro uživatelské rozhraní. Vznikl na bázi knihovny *React form builder,* jejíž vývoj byl ukončen před 3 lety. Od té doby se autor snaží knihovnu dále rozvíjet. Patrné jsou ovšem již zastaralé postupy vývoje původní knihovny, které nebyli doposud odstraněny. Příkladem je zastaralá dokumentace, nestrukturovaný kód a mapování hlavních komponent na *HTML* elementy pomocí *react-dom*[11] ve všech ukázkách použití knihovny.

3.2.6 QFormBuilder

Knihovna *QFormBuilder* nabízí řešení problematiky tvorby webových dynamických formulářů napsané v programovacím jazyce *JavaScript* využívající framework *Vue.js* a *Quasar*[12]. Na serveru *Github.com* má v současné době 43 hvězdiček, 2 otevřená a 1 uzavřený nález. Knihovna je vyvíjena zdarma jako open-source pod licencí *MIT*.



Obrázek 4 QFormBuilder

Nástroj nabízí komponentu pro uživatelskou tvorbu formuláře pomocí *drag and drop* uživatelského rozhraní. Knihovna neřeší ukládání ani vykreslování dat formuláře a přenechává tak tuto odpovědnost vývojáři. Ten má možnost vytvořit vlastní vykreslování formuláře, neboť knihovna exportuje jednotlivé prvky formuláře jako samostatné komponenty. Uživatel má na výběr z 15 různých formulářových prvků. Každý prvek nabízí možnost editace, duplikace

[11] Knihovna zpřístupňující specifické metody *React.js* pro *DOM* využitelné na nejvyšší úrovni aplikace. Nejčastěji využito pro vykreslení *top-level React.js* komponenty do *DOM*.

[12] Framework pro tvoření uživatelského rozhraní ve Vue.js

a smazání. Prvky mohou být shlukovány do skupin, protože jsou k dispozici 2 layout prvky pro vizuální oddělení formulářových prvků. Knihovna navíc nabízí možnost základní validace jednotlivých prvků, tedy nedovolí odeslání formuláře, pokud je nějaký z jeho prvků označen jako povinný a zároveň jej uživatel nevyplnil. Dokumentace knihovny je strukturovaná, obsahuje návod na instalaci, základní i rozšířené příklady užití a popis *API*.

Mezi výhody tohoto nástroje patří uživatelsky přívětivé prostředí, kompaktní grafický návrh, stručná a jasná dokumentace a specializované formulářové prvky jako např. prvek pro přidání údajů platební karty, telefonního čísla, adresy nebo jména. Vývojář ocení možnost správy formulářových dat při integraci do vlastní aplikace.

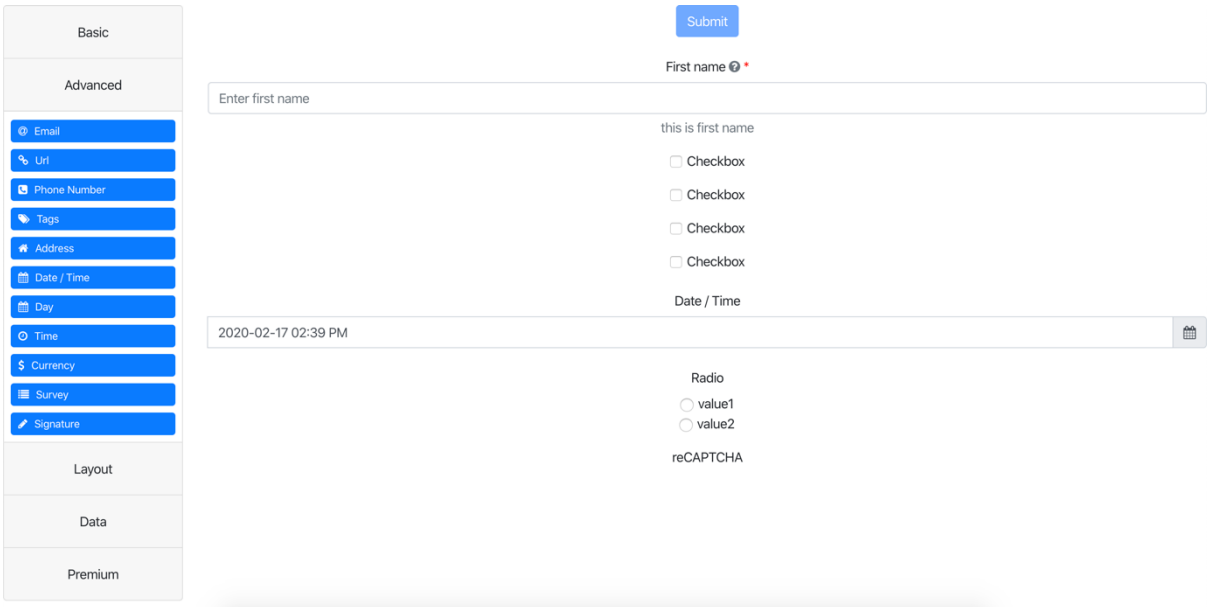
Nevýhodou knihovny z hlediska využitých technologií je použitý framework *Vue.js*. Při integraci do *React.js* aplikace je zapotřebí využití nástroje třetí strany, který se postará o bezproblémovou integraci obou knihoven. Obecně se tento postup nedoporučuje, protože takto slepované řešení je náročnější na zdroje a mnohdy netestovatelné automatizovanými testy. Z hlediska funkcionality je nevýhodou absence komponenty pro vykreslení formuláře a absence některých základních formulářových prvků jako např. pole pro zadání čísla nebo vícenásobný výběr. Nástroj nelze graficky přizpůsobit dle systému do kterého je integrován. Hrozí riziko neočekávaných chyb, protože knihovna není pokryta jednotkovými ani end-to-end testy.

Knihovna *QFormBuilder* je zdařilé řešení tvorby dynamických webových formulářů založené na frameworku *Vue.js*. Využitím frameworku *Quasar* je knihovna graficky celistvá a uživatelsky přívětivá. Knihovna je využitelná v širokém spektru případů užití, neboť zachovává volnost vývojáři při ukládání a správě formulářových dat. Při integraci nástroje do prostředí založeném na *React.js* musí vývojář přijmout výše zmíněná negativa, a navíc spravovat dvě technologicky rozdílná prostředí.

3.2.7 React-formio

React-formio je knihovna využívaná pro zobrazení formuláře vytvořeného na serveru *Form.io* v prostředí *React.js.* Knihovna je vyvíjena pomocí technologií *JavaScript* a *React.js* jako open-source pod licencí *MIT.* Na serveru *Github.com* je hodnocena 92 hvězdičkami a disponuje aktivní komunitou přispěvovatelů. Nástroj je aktuálně ve verzi 4.2.5 a má 67 otevřených a 95 uzavřených nálezů. Vývojovým týmem této knihovny je *Form.io*.

Form.io je *enterprise-level* platforma pro správu dat ve formě formulářů a *API* určená pro vývojáře, kteří vytvářejí vlastní komplexní aplikace podnikových procesů založených na formulářích.[30] Tato *cloud PaaS* služba nabízí vývojáři možnost definice formuláře pomocí *drag and drop* uživatelského rozhraní, které je poté zpřístupněno autorizovaným uživatelům pomocí *web API endpointu*. *React-formio* na základě tohoto *endpointu* vykreslí formulář v klientské aplikaci. Kromě komponenty pro vykreslení nabízí tato knihovna i komponentu pro *drag and drop* definici formuláře, která funkčně kopíruje výše zmíněnou *cloudovou* službu pro tvoření formuláře. Nástroj je proto schopen umožnit uživateli vytvořit formulář, bez potřeby využití *cloudového* nástroje *Form.io.* Přestože je *react formio* vyvíjen jako open-source, pro fungování musí být zpřístupněna komunikace s platformou *Form.io*, která je zpoplatněná. Z hlediska funkcionality nabízí knihovna 24 formulářových prvků, ze kterých uživatel tvoří formulář pomocí uživatelského rozhraní *drag and drop*. Každý prvek lze editovat, mazat a je přístupná podpora pro nastavení základních i rozšířených validací. Existuje možnost shlukovat prvky do skupin pomocí specializovaných prvků. Dokumentace je podrobná a nabízí tutoriály podle využívané technologie.



Obrázek 5 react-formio

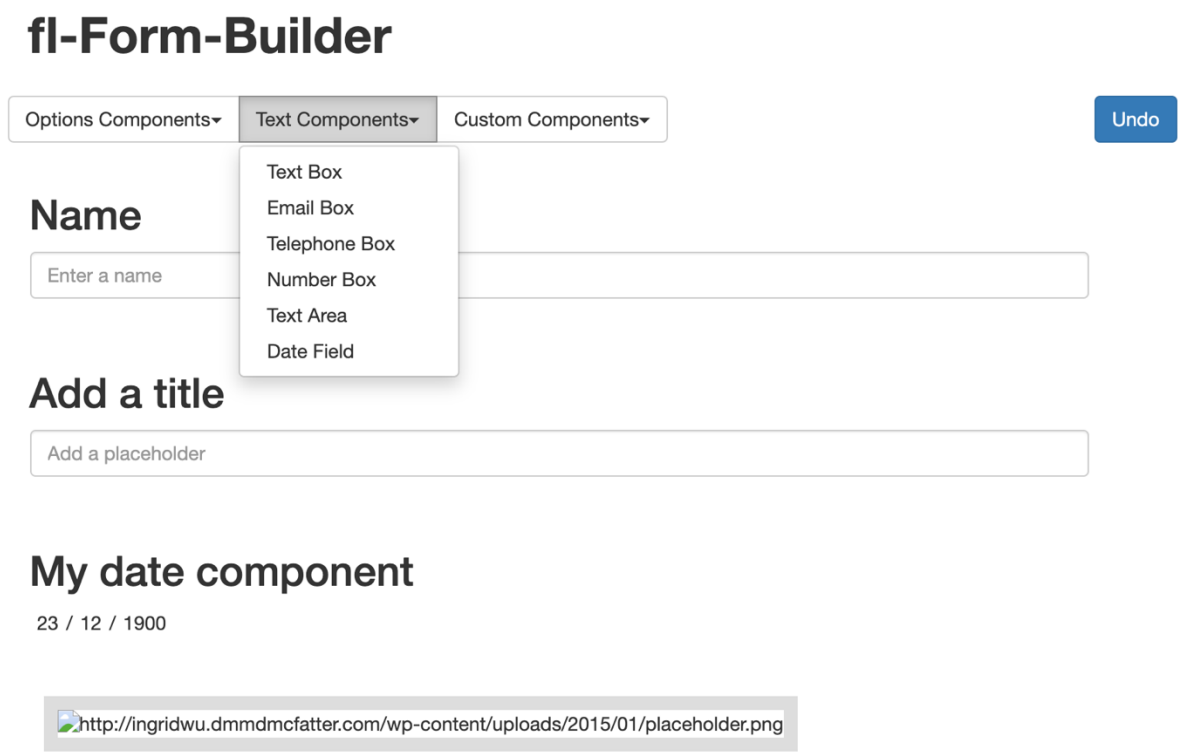
Výhodou tohoto nástroje je široká škála nabízených formulářových prvků a možnost úpravy každého z nich na úrovni atributů, validace nebo např. možnosti podmíněného vykreslení na základě definované podmínky. Integrace s platformou *Form.io* zajistí správu všech formulářů na jednom místě s možností nastavení práv přístupů k *API* každého formuláře na základě skupin.

Nevýhodou nástroje je vyšší cena, která se pohybuje v rozmezí $0 až $250 za měsíc podle nabízených funkcí, přičemž nejnižší tarif zdarma nabízí jen velmi limitovanou funkcionalitu s omezením *API* dotazů na server *Form.io*. Běžný uživatel se snadno ztratí v úpravě jednotlivých formulářových prvků, kvůli vysokému počtu nabízených funkcí a jejich funkčnímu popisu, který je zaměřený hlavně na vývojáře.

React-formio je nástroj určený pro sofistikované vytváření a správu formulářů, který je zaměřený na větší firmy a podniky. Menší týmy a vývojáře nástroj odradí cenou a nutností integrace platformy *Form.io*.

3.2.8 Fl-form-builder

Fl-form-builder je nástroj pro tvorbu dynamických webových formulářů napsaný v jazyce *Javascript*. Knihovna je vyvíjena jako *open-source* pod licencí *MIT*. Na serveru *Github.com* je hodnocena 12 hvězdičkami, má 4 uzavřené nálezy a aktuálně je ve verzi 1.4.2.



Obrázek 6 fl-form-builder

V dokumentaci autor uvádí, že nástroj je psaný v tzv. *VanillaJS*[13]. Po bližším zkoumání zdrojového kódu zjistíme, že knihovna nabízí typovou definici pomocí knihovny *prop-types*, která se využívá pro otypování vlastností komponent v *React.js*. Uživatelské rozhraní a logika aplikace jsou psané pomocí *React.js* a dalších pomocných knihoven. S využitím *Babel*[14] je výsledné řešení zkompilované do *VanillaJS*. Z funkčního hlediska knihovna obsahuje komponentu pro tvoření formuláře a možnost importu a exportu *JavaScript* objektu s popisem prvků vytvářeného formuláře. Nenabízí tedy komponentu pro vykreslení formuláře a přenechává tuto odpovědnost vývojáři. Knihovna poskytuje 11 formulářových prvků pro tvoření formuláře. Každý prvek nabízí základní editaci parametrů a nastavení volitelnosti, tedy nejzákladnější validaci. Uživatel má k dispozici tři rozbalovací tlačítka s volbou prvků pro přidání do formuláře. Prvky ve formuláři nabízejí možnost smazání nebo roztřídění pomocí *drag and drop*. Dokumentace je stručná a obsahuje základní příklady použití a popis typů pro vlastní definici formulářových prvků.

Výhodou nástroje je jeho jednoduché uživatelské použití. Knihovna zaznamenává akce při tvoření formuláře a umožňuje uživateli mazat poslední pomocí tlačítka *undo.* Vývojáři ocení typovou definici *React.js* komponent, existenci jednotkových testů a možnosti rozšíření o vlastní formulářové prvky pomocí definovaného *API* v dokumentaci.

Pro integraci knihovny do vlastního systému je třeba doinstalovat povinné závislosti, které knihovna obsahuje, tedy *react* a *react-dom* ve verzi nižší než *15.5.0* (představená v dubnu 2017). Dokumentace knihovny neobsahuje informace o instalaci. Z tohoto důvodu a zastaralosti použitých technologií knihovny je instalace složitá a probíhá postupným odstraňováním chyb při běhu programu. Z hlediska funkčnosti knihovna nabízí jen omezené množství formulářových prvků a chybí nástroj pro vykreslení formuláře. Prvky formuláře nenabízí možnost duplikace a editace atributů je v porovnání s ostatními analyzovanými knihovnami nejomezenější.

Fl-form-builder nabízí řešení využitelné, ze své podstaty, v jakékoliv moderní technologii založené na programovacím jazyce *JavaScript*. Kvůli neaktuálním verzím využívaných

---

[13] Humorné označení pro fiktivní framework, obsahující jen jazyk *JavaScript* a jeho funkce. Využíván k vyzdvihnutí často neopodstatněné potřeby vývojářů využívat knihovny a frameworky na problémy, které lze řešit pouze *JavaScriptem.*

[14] *JavaScript* kompilátor. Převádí vyšší verze jazyka JavaScript na verzi podporovanou nejvíce prohlížeči.

balíčků a nutnosti instalace zastaralých závislostí přichází knihovna o tuto nepopiratelnou výhodu.

3.3 Výsledek analýzy

Po vývojářské stránce se jako nejslibnější řešení jeví knihovna *formBuilder,* která získala 47,8 bodů z celkových 70, tedy v přepočtu 68,3 % a stala se i kompromisním řešením dle *TOPSIS* analýzy. Další v pořadí se umístila komerčně nabízená knihovna *react-formio* s 55,4 %. Zbylé knihovny se pohybují na úrovni 42 %. Výsledek vývojářské analýzy můžeme interpretovat tak, že řešení, které se umístilo na prvním místě se slabě blíží k ideálnímu řešení, které by bylo hodnoceno 70 body, tedy 100 %. Přehled výsledků knihoven je zobrazen v grafu 2. Podrobněji je vývojářské hodnocení rozepsáno v tabulce 5.

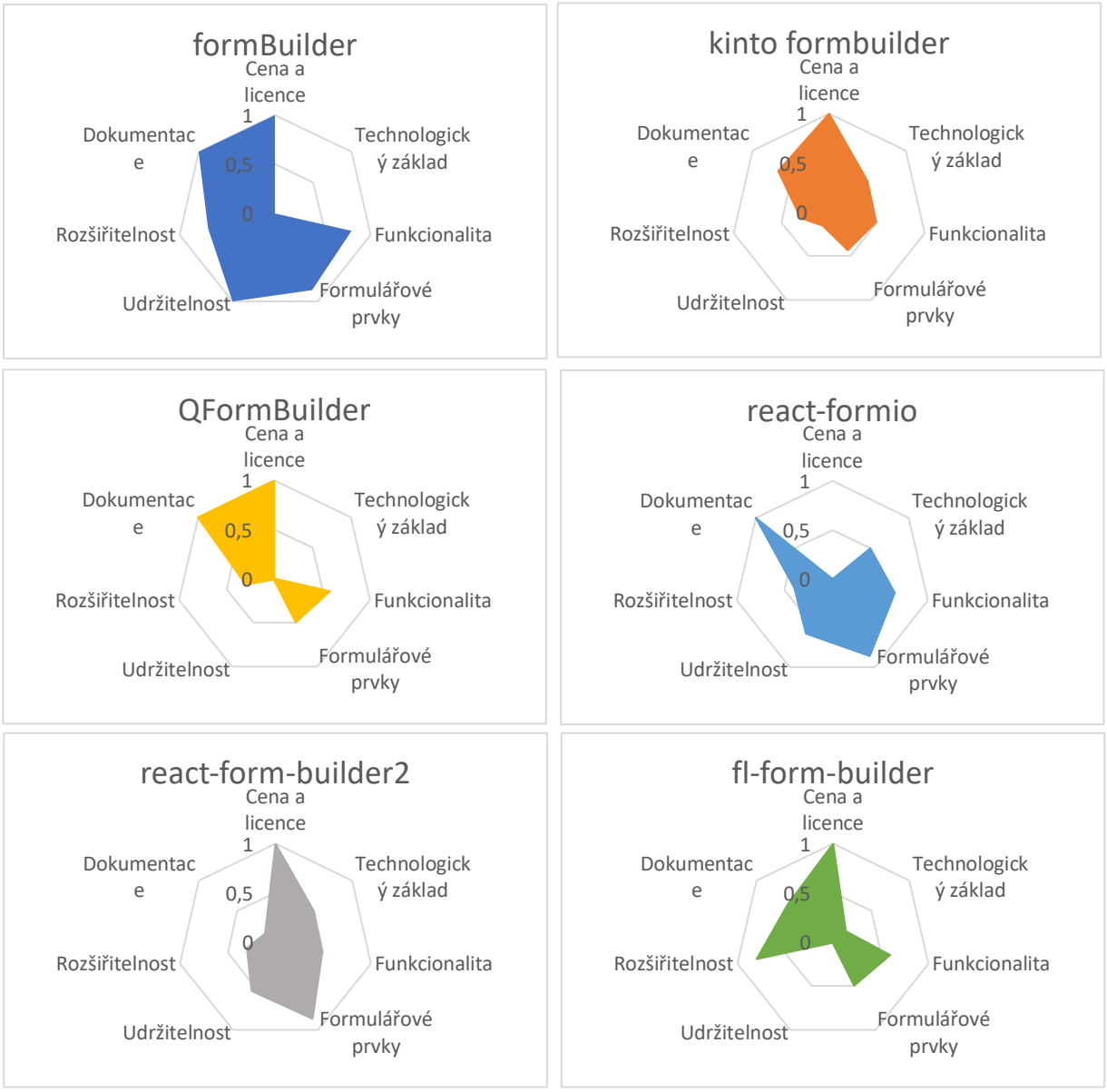
Tabulka 5 Výsledek analýzy dle vývojářských kritérií

Symbol	Kritérium	Hodnotící sekce	Max. bodů	formBuilder	kinto formbuilder	react-form-builder2	QFormBuilder	react-formio	fl-form-builder
V1	Cena a licence	užití zdarma	5	✓	✓	✓	✓	×	✓
		svobodná licence	5	✓	✓	✓	✓	✓	✓
sekce celkem			10	10	10	10	10	5	10
V2	Technologický základ	využitelné v React.js	3	✓	✓	✓	✓	✓	✓
		napsané v React.js	3	×	✓	✓	×	✓	×
		API v. 16.8.0 a vyšší	1	×	×	×	×	×	×
		TypeScript	1	×	×	×	×	×	×
		typová definice	2	×	×	×	×	×	✓
sekce celkem			10	3	6	6	3	6	5
V3	Funkcionalita	import schématu	1	×	×	×	×	×	✓
		export schématu	1	✓	×	×	×	×	✓
		form renderer	1	✓	✓	✓	×	✓	×
		přizpůsobení	1	×	×	×	×	×	×

		validace – základní	1	✓	×	×	✓	✓	✓
		validace – rozšířená	1	×	×	×	×	✓	×
		shlukováni prvků	1	×	×	×	✓	✓	×
		duplikace prvku	1	✓	×	×	✓	×	×
		vymazání prvku	1	✓	✓	✓	✓	✓	✓
		editace prvku	1	✓	✓	✓	✓	✓	✓
sekce celkem			10	6	3	3	5	6	5
V4	Formulářové prvky	Button	0,6	✓	×	×	×	✓	×
		Checkbox	0,6	✓	✓	✓	✓	✓	✓
		Checkbox Multiple	0,6	✓	✓	✓	✓	✓	✓
		Date	0,6	✓	✓	✓	✓	✓	✓
		File upload	0,6	✓	×	✓	✓	✓	×
		Header	0,6	✓	×	✓	×	✓	×
		Hidden input	0,6	✓	×	×	×	✓	×
		Number	0,6	✓	×	✓	×	✓	✓
		Paragraph	0,6	✓	×	✓	×	✓	×
		Radio	0,6	✓	✓	✓	✓	✓	✓
		Range	0,6	×	×	✓	×	×	×
		Rating	0,6	×	×	✓	×	×	×
		Select	0,6	✓	✓	✓	✓	✓	✓
		Select Multiple	0,6	✓	×	✓	×	✓	×
		Text field	0,6	✓	✓	✓	✓	✓	✓
		Text area	0,6	✓	✓	✓	✓	✓	✓
sekce celkem			10	8,8	4,4	8,8	5	8,8	5
V5	Udržitelnost	vývoj v posl. 3 m.	4	✓	×	✓	×	✓	×
		github stars*	3	1800	538	131	42	92	12
		přispěvovatelé**	3	35	12	8	2	22	2
sekce celkem			10	10	1,9	4,9	0,2	6	0,2
V6	Rozšiřitelnost	api	5	✓	×	×	×	×	✓

		unit testy	2	×	×	×	×	✓	✓
		end-to-end testy	2	×	×	×	×	×	×
		open-source	1	✓	✓	✓	✓	✓	✓
sekce celkem			10	6	1	1	1	3	8
V7	Dokumentace	strukturovaná dokumentace	2	✓	✓	✓	✓	✓	✓
		instalační pokyny	2	✓	✓	×	✓	✓	×
		příklady – základní	2	✓	✓	×	✓	✓	✓
		příklady – pokročilé	2	✓	×	×	✓	✓	×
		popis api	2	✓	×	×	✓	✓	✓
sekce celkem			10	10	6	2	10	10	6
celkem			**70**	**47,8**	**29,3**	**32,7**	**29,2**	**38,8**	**34,2**

* 1 hvězdička = 1/600 bodu
** 1 přispěvovatel = 3/35 bodu

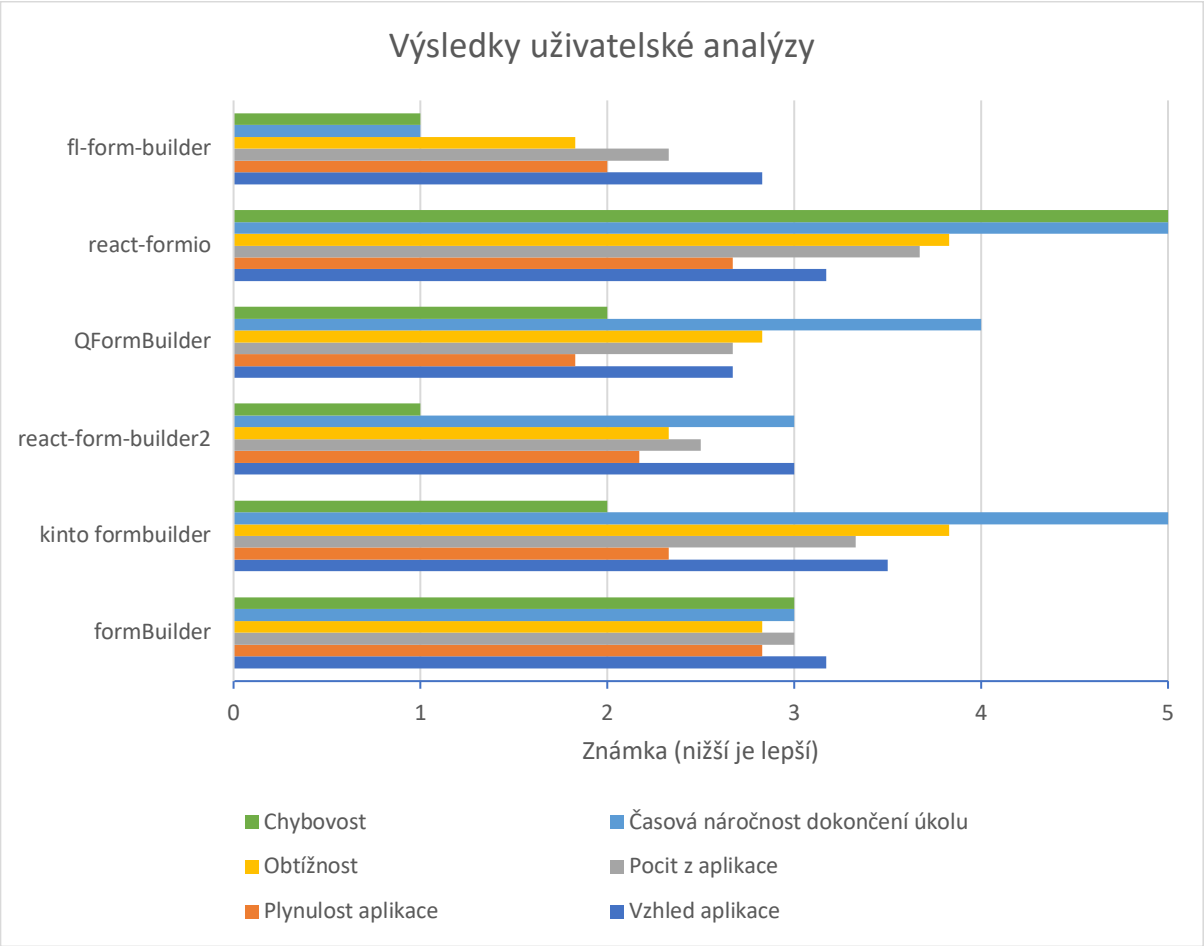


Graf 2 Výsledky vývojářského hodnocení knihoven

Uživatelské testování je podrobně rozepsané v tabulce 6. Známkové ohodnocení uživatelských kritérií je zobrazeno na grafu 3. Pokud bychom srovnali knihovny podle výsledků jednotlivých kritérií, vítězem by byla knihovna *fl-form-builder* následovaná *react-form-builder2.* Dva z šesti účastníků odmítli pokračovat v řešení zadaného úkolu s knihovnou *react-formio,* proto je toto řešení hodnoceno nejhůře.

Zajímavé je, že se vítězové vývojářské analýzy umístily na posledních příčkách v hodnocení uživatelů. To může být způsobeno tím, že jsou tyto knihovny určeny pro jiné uživatele, než byla cílová skupina *UX* testování, kterou zvolil autor. Dalším důvodem může být absence uživatelského testování při vývoji těchto nástrojů.

Výsledky analýzy ukazují, že v současné době neexistuje ideální řešení, které by splňovalo požadavky z vývojářského a zároveň uživatelského pohledu, respektive existují jen řešení, které zaostávají v jedné nebo druhé oblasti. Analýza primárně odhalila slabé a silné aspekty jednotlivých knihoven, které poslouží k vytyčení minimálních požadavků a budou zohledněny při tvorbě vlastního řešení.



Graf 3 Výsledky uživatelského hodnocení knihoven

Tabulka 6 Výsledek analýzy uživatelského testování

Symbol	Kritérium	Účastníci	formBuilder	kinto formbuilder	react-form-builder2	QFormBuilder	react-formio	fl-form-builder
U1	Vzhled aplikace	A	3	4	3	1	3	2
		B	2	4	3	1	3	4
		C	5	3	3	3	3	4
		D	3	3	3	4	5	2
		E	2	5	4	3	2	3
		F	4	2	2	4	3	2
aritmetický průměr			3,17	3,5	3	2,67	3,17	2,83
U2	Plynulost aplikace	A	3	3	2	1	2	2
		B	1	1	1	1	1	2
		C	4	2	2	2	2	1
		D	3	3	3	3	5	2
		E	2	4	3	3	4	4
		F	4	1	2	1	2	1
aritmetický průměr			2,83	2,33	2,17	1,83	2,67	2
U3	Pocit z aplikace	A	2	3	2	2	2	2
		B	2	5	2	2	3	3
		C	5	4	3	4	5	2
		D	3	2	4	3	5	2
		E	2	4	2	2	3	3
		F	4	2	2	3	4	2
aritmetický průměr			3	3,33	2,5	2,67	3,67	2,33
U4	Obtížnost	A	2	5	1	2	3	1
		B	2	5	2	2	2	2
		C	4	2	3	3	5	1
		D	4	4	4	4	5	2
		E	2	5	1	2	4	3
		F	3	2	3	4	4	2
aritmetický průměr			2,83	3,83	2,33	2,83	3,83	1,83
U5	Množství času potřebného k dokončení úkolu	A	3:36	3:56	3:54	6:54	10:13	3:05
		B	3:32	8:21	4:01	2:59	3:39	5:15
		C	6:25	4:28	5:56	4:45	*	2:57
		D	6:02	8:07	6:15	9:56	**	4:35
		E	3:08	2:51	2:32	1:41	2:19	1:55
		F	3:27	3:17	3:32	4:20	5:24	3:24

aritmetický průměr			4:05	5:10	4:21	5:05	5:23	3:31
U6	Počet chyb	A	4	3	1	4	3	1
		B	2	7	0	0	1	4
		C	6	1	6	2	10*	1
		D	5	5	5	7	13**	3
		E	2	0	1	0	1	2
		F	2	3	3	5	3	5
aritmetický průměr			3,5	3,17	2,67	3	5,17	2,67

* účastník odmítl v nástroji úkol dokončit po 4:05 minutách

** účastník odmítl v nástroji úkol dokončit po 4:28 minutách

4 Návrh a realizace vlastního řešení

Kapitola popisuje návrh a implementaci vlastního řešení – knihovny *React Form Architect*. V úvodu jsou definovány funkční a nefunkční požadavky. Dále je k nahlédnutí logický rámec aplikace a scénáře jejího užití. V druhé polovině kapitoly je specifikována architektura řešení, jsou přiblíženy použité technologie a je popsána implementace jednotlivých funkčních celků, které knihovna nabízí, její dokumentace a způsob testování. Konec kapitoly obsahuje komparaci knihovny s jejími konkurenty z kapitoly 3.

4.1 Funkční požadavky

V kapitolách 2 a 3 byly provedeny rešerše a analýzy, při kterých autor identifikoval základní funkční požadavky pro realizaci vlastního řešení, které dále rozšiřuje a přidává vlastní požadavky hlavně dle potřeb *UniCatDB*. Funkční požadavky jsou vypsány v tabulce 7.

Tabulka 7 Funkční požadavky

Kód	**Požadavek**
F1	Sestavení formuláře pomocí Drag and Drop funkcionality
F2	Export a import schématu formuláře pomocí JSON
F3	Úprava atributů formulářových prvků
F4	Definice validačních pravidel formulářových prvků
F5	Duplikace formulářového prvku
F6	Mazání formulářového prvku
F7	Změna pořadí formulářových prvků
F7	Shlukování prvků do skupin
F8	Změna pořadí skupin
F9	Přejmenování skupiny
F10	Mazání skupiny
F11	Přesouvání prvků mezi skupinami
F12	Vykreslení formuláře
F13	Získání validovaných dat z formuláře

F14	Formulářový prvek pro práci se stromově strukturovanými daty
F15	Vykreslení stromově strukturovaných dat
F16	Vyhledání uzlu ve stromově strukturovaných datech
F17	Výběr obsahu uzlu stromově strukturovaných dat

4.2 Nefunkční požadavky

Nefunkční požadavky vychází hlavně z potřeb *UniCatDB* a z identifikovaných slabin analyzovaných řešení při vývojářské analýze existujících knihoven na tvorbu dynamických formulářů z kapitoly 3. Seznam požadavků je zobrazen v tabulce 8.

Tabulka 8 Nefunkční požadavky

Kód	**Požadavek**
N1	Publikace pod MIT licencí
N2	Publikace ve formě NPM modulu
N3	Rozšiřitelnost a modifikovatelnost
N4	Udržitelnost

Nefunkční požadavek *N3* – Rozšiřitelnost a modifikovatelnost – popisuje schopnost aplikace být rozšířenou o další formulářové prvky a možnost úpravy tématu, kterým se řídí stylování aplikace. Nefunkční požadavek *N4* – Udržitelnost – definuje potřebu oddělit funkcionalitu aplikace do na sebe nezávislých celků a pokrytí kódu jednotkovými testy.

4.3 Logický rámec

Tabulka 9 Logický rámec projektu

Cíl	**Objektivně ověřitelné ukazatele**	**Způsob ověření**	**Předpoklady**
Snazší, efektivnější a levnější způsob sběru přírodovědných velkých dat na webu	Nárůst počtu online definovaných a dále zpracovaných velkých dat alespoň v 1 oboru přírodních věd	Statistiky sběru dat konkrétního přírodovědného oboru Výroční zprávy a souhrny	
Účel	**Objektivně ověřitelné ukazatele**	**Způsob ověření**	**Předpoklady**
Možnost definice strukturálně odlišných dat bez potřeby zapojení specializované osoby.	Existuje alespoň 1 projekt, který nabízí uživateli možnost vlastní definice heterogenních dat	Dokumentace nebo vědecký článek projektu, který popisuje způsob, jak uživatelsky tvořit data různého typu	Projekt je dostupný online a má uživatelské rozhraní. Vzájemná integrace databáze a výstupu uživatelsky definovaných dat Povědomí vědecké společnosti o existenci projektu
Výstupy	**Objektivně ověřitelné ukazatele**	**Způsob ověření**	**Předpoklady**
Knihovna pro tvorbu dynamických webových formulářů	Publikovaný balíček v databázi npm Online repozitář v GitHub databázi	Databáze npmjs.com Databáze Github.com	Užití všeobecně uznávaných postupů pro tvorbu software knihovny Strukturovaná dokumentace Open source licence
Činnosti	**Zdroje**	**Časový rámec**	**Předpoklady**
Tvorba knihovny pro tvorbu dynamických formulářů Definice klíčových vlastností Uživatelské testování nástrojů (usability testing) Rešerše a analýza dostupných nástrojů pro uživatelskou definici heterogenních dat Analýza způsobů zpracování přírodovědeckých dat	Relevantní vzorek lidí pro uživatelské testovaní Online databáze portálů pro správu přírodovědných dat s popisem jejich struktury Online databáze nástrojů nabízející požadovanou funkcionalitu Strukturovaný postup uživatelského testování Pracovní stanice s přístupem k internetu	12/2020 . . . . 01/2020	Zodpovědný přístup vývojového týmu Kvalitní analýza a definice klíčových vlastností Popis práce odpovídající akademickým standardům

4.4 Scénáře použití

PŘIDAT FORMULÁŘOVÝ PRVEK DO SKUPINY

Přiřazení formulářového prvku do právě aktivní skupiny.

Základní scénář

1. Uživatel přidává formulářový prvek do skupiny přetažením ze sady formulářových prvků do oblasti formulářového editoru.

UPRAVIT ATRIBUTY FORMULÁŘOVÉHO PRVKU

Upravení atributů formulářového prvku.

Vstupní podmínky

- Existuje formulářový prvek ve formulářovém editoru.

Základní scénář

1. Uživatel volí akci úpravy formulářového prvku.
2. Systém zobrazuje editor formulářového prvku.
3. Uživatel příslušné atributy formulářovému prvku.
4. Uživatel potvrzuje úpravu změn.

PŘIDAT VALIDACE FORMULÁŘOVÉHO PRVKU

Přiřazení validačních pravidel a jim odpovídajících chybových hlášek formulářového prvku.

Vstupní podmínky

- Existuje formulářový prvek ve formulářovém editoru.

Základní scénář

1. Uživatel volí akci úpravy formulářového prvku.
2. Systém zobrazuje editor formulářového prvku.
3. Uživatel nastavuje validační pravidla a k nim příslušné chybové hlášky formulářovému prvku.
4. Uživatel potvrzuje úpravu změn.

DUPLIKOVAT FORMULÁŘOVÝ PRVEK

Vytvoření kopie formulářového prvku.

Vstupní podmínky

- Existuje formulářový prvek ve formulářovém editoru.

Základní scénář

1. Uživatel volí akci duplikace formulářového prvku.

SMAZAT FORMULÁŘOVÝ PRVEK

Odstranění formulářového prvku ze skupiny.

Vstupní podmínky

- Existuje formulářový prvek ve formulářovém editoru.

Základní scénář

UŽIVATEL VOLÍ AKCI SMAZÁNÍ FORMULÁŘOVÉHO PRVKU.PŘEŘADIT PRVEK DO JINÉ SKUPINY

Přesun prvku z právě aktivní skupiny do jiné.

Vstupní podmínky

- Existuje formulářový prvek ve formulářovém editoru.
- Existují alespoň 2 skupiny.

Základní scénář

1. Uživatel volí akci změny skupiny formulářového prvku.
2. Systém zobrazuje seznam skupin.
3. Uživatel volí cílovou skupinu.

ZMĚNIT POŘADÍ PRVKU VE SKUPINĚ

Změna pořadí prvků v právě aktivní skupině.

Vstupní podmínky

Existují alespoň 2 formulářové prvky ve stejné skupině ve formulářovém editoru.

Základní scénář

1. Uživatel tažením přesouvá formulářový prvek na určenou pozici.
2. Systém mění pořadí taženého prvku a prvku na aktuální pozici kurzoru.

PŘIDAT SKUPINU

Přidání nové skupiny do seznamu skupin.

Základní scénář

1. Uživatel volí možnost přidání nové skupiny.

PŘEJMENOVAT SKUPINU

Změna názvu právě aktivní skupiny.

Základní scénář

1. Uživatel volí možnost přejmenování skupiny.
2. Systém zobrazí textové pole s před-vyplněným názvem právě aktivní skupiny.
3. Uživatel mění název skupiny a potvrzuje uvedené změny.

SMAZAT SKUPINU

Odebrání právě aktivní skupiny ze seznamu skupin.

Vstupní podmínky

Existují alespoň 2 skupiny.

Základní scénář

1. Uživatel volí možnost smazání skupiny.

2. Systém upozorňuje uživatele, že smazáním skupiny přijde o všechny formulářové prvky, které skupina obsahuje.
3. Uživatel potvrzuje smazání skupiny.

ZMĚNIT POŘADÍ SKUPIN

Změna pořadí skupiny.

Vstupní podmínky

Existují alespoň 2 skupiny.

Základní scénář

1. Uživatel tažením přesouvá skupinu na určenou pozici.
2. Systém mění pořadí tažené skupiny a skupiny na aktuální pozici kurzoru

IMPORTOVAT SCHÉMA

Import schématu, tedy předpisu formulářových prvků a skupin.

Základní scénář

1. Uživatel volí možnost přenosu dat.
2. Systém zobrazuje panel s možnostmi přenosu dat.
3. Uživatel volí možnost „*Import*“ a vybírá lokální soubor s koncovkou *JSON*.
4. Systém validuje strukturu vloženého souboru.
5. Systém vytváří formulářové prvky a skupiny dle validovaných dat vloženého schématu.

EXPORTOVAT SCHÉMA

Export schématu ve formě *JSON* souboru nebo ve formě posloupnosti znaků zkopírovaných do schránky.

Základní scénář

1. Uživatel volí možnost přenosu dat.

2. Systém zobrazuje panel s možnostmi přenosu dat.
3. Uživatel volí možnost „*Export JSON*“.

4. Systém vytváří *JSON* soubor ze schématu a posílal jej uživateli.

VYKRESLIT PROSTŘEDÍ PRO TVORBU FORMULÁŘE

Zobrazení formulářového editoru a sady formulářových prvků.

Základní scénář

1. Uživatel spouští aplikaci pro tvorbu formuláře.

VYKRESLIT FORMULÁŘ DLE SCHÉMATU

Zobrazení formulářového editoru a sady formulářových prvků.

Vstupní podmínky

Existuje formulářové schéma.

Základní scénář

1. Systém podle schématu vykresluje formulářové prvky, vytváří validační schéma a výsledek balí formulářovým kontextem.

VYKRESLIT STROM

Zobrazení stromově strukturovaných dat.

Základní scénář

1. Systém vykresluje stromově strukturovaná data ve formě uzlů na obrazovku.

DEFINOVAT NOVÝ FORMULÁŘOVÝ PRVEK

Rozšíření sady formulářových prvků o další prvek.

Základní scénář

1. Integrátor aplikace definuje nový formulářový prvek a předkládá ho na vstup aplikace pro tvorbu formuláře.
2. Systém validuje předaný prvek na vstupu a slučuje jej s výchozí sadou formulářových prvků.

VYPLNIT FORMULÁŘ

Naplnění formulářových prvků hodnotami.

Základní scénář

1. Uživatel vyplňuje formulář.

ZÍSKAT VALIDOVANÁ DATA Z FORMULÁŘE

Získání správně vyplněných dat formuláře.

Vstupní podmínky

Existuje vyplněný formulář.

Základní scénář

1. Systém validuje hodnoty jednotlivých formulářových prvků podle validačního schématu a předkládá je na výstup aplikace pro vykreslení formuláře.

VYHLEDAT UZEL STROMU

Vyhledání konkrétního uzlu stromu podle hledaného výrazu.

Základní scénář

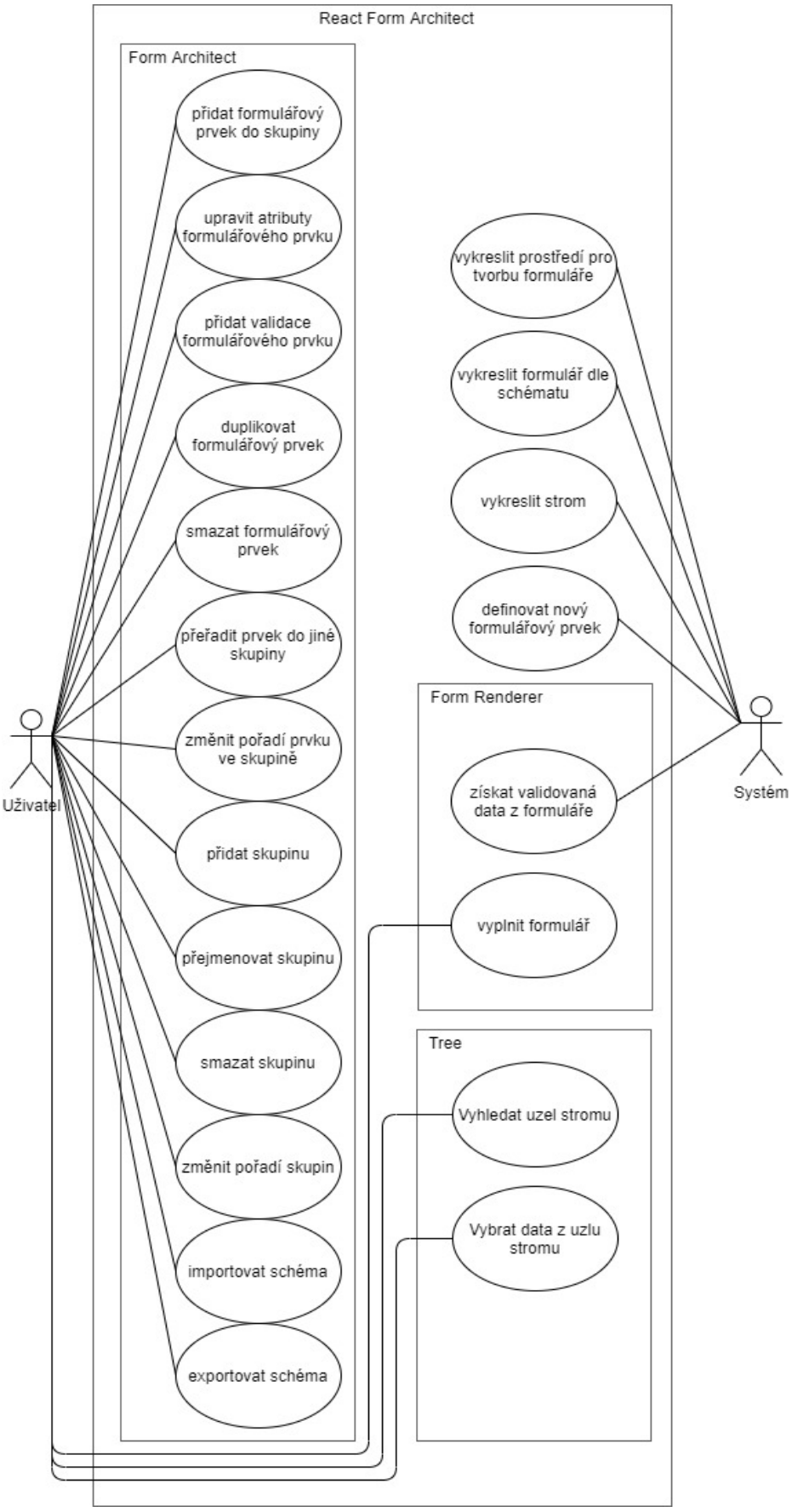
1. Uživatel zadává hledaný výraz do textového pole.
2. Systém prochází stromově strukturovaná data a porovnává hledaný výraz s obsahem jednotlivých uzlů.
3. Systém zobrazuje nalezené uzly.

VYBRAT DATA Z UZLU STROMU

Získání dat specifického uzlu stromu.

Základní scénář

1. Uživatel kliká na uzel stromu.
2. Systém předkládá data uzlu na výstup aplikace pro vykreslení stromu.



Obrázek 7 Diagram případů užití RFA

4.5 Technologie

Vlastní řešení s názvem *React Form Architect* (dále *RFA*) je zhotoveno jako webová front-end aplikace napsaná v jazyce *JavaScript* s využitím nadstavby *TypeScript*[15] pro statickou kontrolu typovosti. O vykreslení *UI* se stará knihovna *React.js*. Vytvořené komponenty jsou psány funkcionálním způsobem a je využit princip „hooků", které *React.js* nabízí. Dizajn aplikace je vytvořen pomocí knihovny *Material-UI*[16] a následuje „Best Practice" dané sbírkou *Material Design*[17]. Ke stylování je využit přístup „*CSS in JS*". Sytém *drag and drop* je zpřístupněn knihovnou *React DnD*[18]. Formulářový kontext je řízen za pomoci *React Hook Form*[19]. Jednotkové testy jsou psány s podporou knihovny *Jest*[20] a uživatelské testy pomocí knihovny *Cypress*[21].

Aplikace je publikovaná jako *npm* modul pod *MIT* licencí za pomoci *rollup.js*[22] a *Babel*[23]. O kontrolu kvality kódu se starají knihovny *ESLint*[24] a *Prettirer*[25]. Knihovna podporuje eliminování nepoužívaného kódu pomocí techniky „*tree shaking*". Verzovaní zajišťuje systém *git* a repositář sídlí na webu *GitHub.com.* Dokumentace knihovny je psaná ve formě *markdown* souborů, které jsou pomocí balíčku *Jekyll*[26] přeloženy do statické *HTML* stránky hostované pomocí *GitHub Pages*[27].

4.6 Architektura RFA

Funkcionalita knihovny je rozdělena do tří hlavních komponent. Komponenta *FormArchitect* vykresluje prostředí pro sestavení formuláře, úpravu atributů formulářových prvků, definici

---

[15] https://www.typescriptlang.org/

[16] http://material-ui.com/

[17] https://material.io/design

[18] https://github.com/react-dnd/react-dnd/

[19] https://react-hook-form.com/

[20] https://jestjs.io/

[21] https://www.cypress.io/

[22] https://rollupjs.org/

[23] https://babeljs.io/

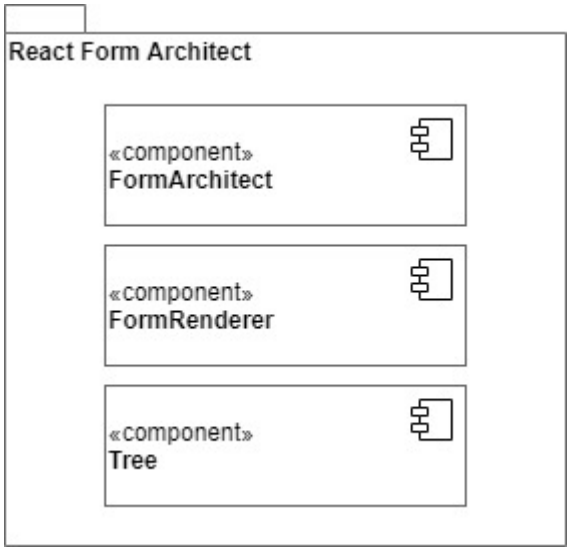
[24] https://eslint.org/

[25] https://prettier.io/

[26] https://jekyllrb.com/

[27] https://pages.github.com/

validačních pravidel, seskupování prvků, exportu a importu schématu ve formě *JSON* souboru. Komponenta *FormRenderer* vykresluje formulář dle schématu, validuje uživatelem vyplněné údaje a případně zobrazuje chybové stavy. Komponenta *Tree* vykresluje stromově strukturovaná data a dovoluje uživateli procházet strom po jednotlivých uzlech a vybrat data k dalšímu použití. Navíc umožňuje i vyhledání uzlu ve stromě. Obrázek 8 zobrazuje digram struktury balíčku *RFA*.



Obrázek 8 Struktura RFA

Komponenty společně tvoří prostředí pro sestavení formuláře, vykreslení formuláře a zobrazení stromově strukturovaných dat. I když je primárním cílem knihovny, aby byly komponenty použity společně, může být každá z nich vynechána, popřípadě nahrazena jiným řešením. Všechny tři komponenty jsou na sobě nezávislé. Předávání dat mezi nimi, je-li potřeba, zajišťuje systém, do kterého je knihovna integrována. Výhodou takhle strukturovaného modulárního přístupu je flexibilita, se kterou si může integrátor knihovny přizpůsobit výsledek podle vlastních potřeb, a udržitelnost, protože každá komponenta má oddělenou logiku a vlastní sadu testů.

4.7 Implementace

Následuje popis implementace knihovny *RFA*. Přiblíženy jsou dílčí celky hlavních komponent, které knihovna poskytuje. Společným prvkem všech komponent je možnost definice tzv. Tématu, který upravuje vzhled komponent a je volitelně přijímaný na vstupech jednotlivých komponent *RFA*.

4.7.1 FormArchitect

Komponenta *FormArchitect* slouží k vytvoření schématu formuláře, který je následně využit dalšími komponentami. Komponenta se skládá z více částí, přičemž je snaha zachovat přehlednost a čistotu designu minimalizací přebytečných *UI* prvků, které by mohli na uživatele působit zmatečným dojmem. Ukázka komponenty s popisem jednotlivých částí je k nahlédnutí na obrázku 9. Po vykreslení komponenty vidí uživatel dvě nejvýznačnější části – sadu formulářových prvků a formulářový editor – doplněné o sekundární části, které rozšiřují funkcionalitu komponenty.

Základem komponenty je pole řídících objektů, které je inicializováno před jejím prvním vykreslením. Každý objekt v poli popisuje jeden formulářový prvek, který může uživatel použít při sestavování formuláře. Definuje název prvku, jeho označení, ikonu a popis v sadě formulářových prvků, jeho atributy a možná validační pravidla. Dále je uveden validační typ a ukazatel na funkci, která slouží k vykreslení prvku, tedy tzv. *React Element.* Atributy jsou vymezeny názvem, typem a hodnotou, přičemž typ slouží k vykreslení patřičného vstupu pro úpravu atributu v editoru formulářového prvku. Validační pravidla jsou uvedena jako názvy jednotlivých validátorů, které jsou implementovány v rámci knihovny. Validační typ je označení datového typu výstupu formulářového prvku ve formuláři, který slouží jako konstruktor při sestavování validačního schématu, jež je blíže přiblížen v oddílu 4.7.2. Komponenta na svém vstupu volitelně přijímá objekty, kterými může integrátor knihovny rozšířit výchozí pole prvků a přidat tak vlastní formulářové prvky, jejich atributy a připojit definované validátory.

Obrázek 9 Popis částí komponenty FormArchitect

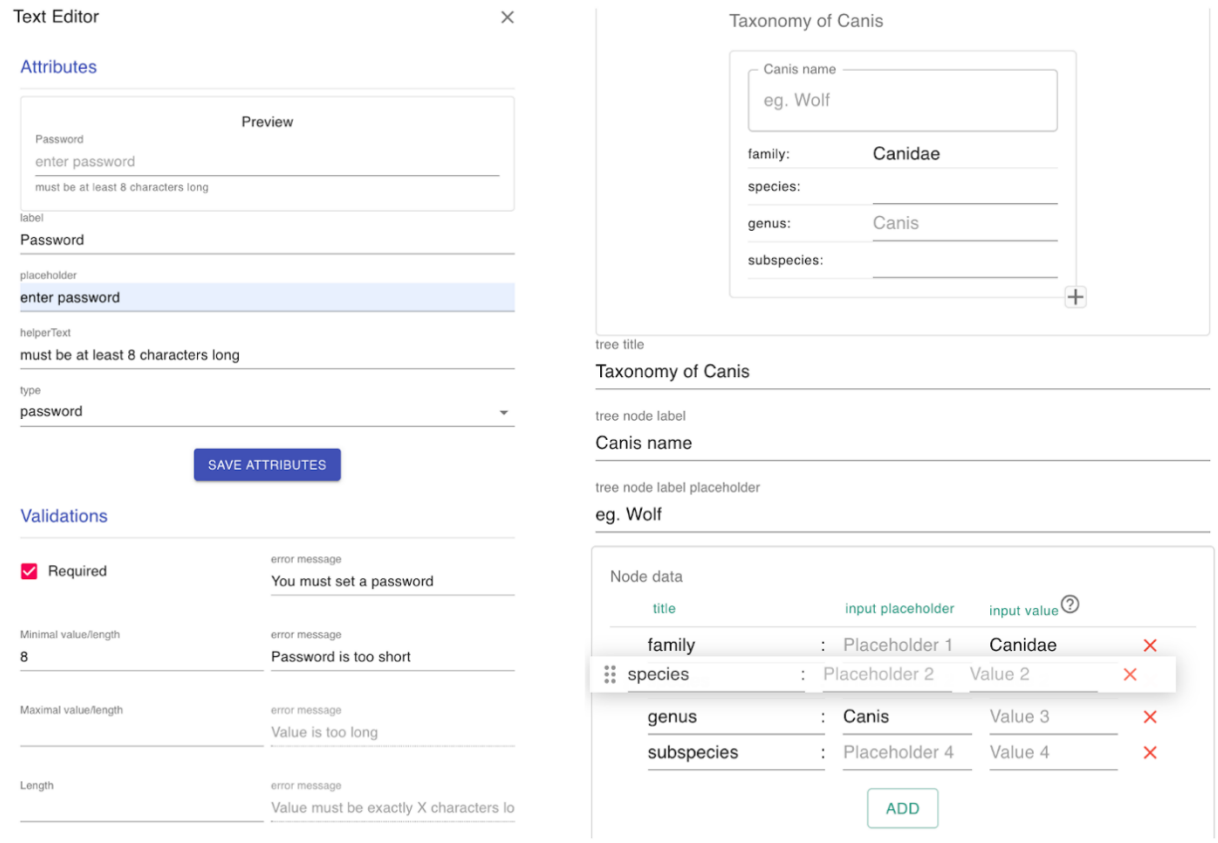
Sada formulářových prvků, ze které uživatel vybírá a přidává jednotlivé prvky do formuláře, je komponenta, kterou – v určité formě – obsahují všechna analyzovaná řešení z kapitoly 3. Během uživatelského testování vyšlo najevo, že tento prvek způsobuje uživatelům značné problémy při sestavování formuláře, a to z důvodu uživatelské neznalosti termínů užívaných k označení formulářových prvků. Proto autor přidal při implementaci k jednotlivým formulářovým prvkům kromě jejich názvu i charakterizující ikony a pomocný text popisující vhodné užití jednotlivých prvků. Dalším identifikovaným problémem bylo samotné přidání prvku ze sady do editoru. Zatímco někteří účastníci testování zkoušeli přidávat prvky pomocí tažení ze sady na určité místo v editoru, jiní se snažili prvek přidat kliknutím myši. Pokud testované řešení nepodporovalo jednu nebo druhou popsanou možnost, uživatel se na místě zasekl a doba plnění úkolu se značně prodloužila. Při implementaci proto autor zpřístupnil obě tyto metody pro přidání prvků do editoru a prvkům přidal zvýraznění při tažení, aby zdůraznil, že jsou ovladatelné metodou *drag and drop.*

Formulářový editor je plocha, která přijímá formulářové prvky a následně je zobrazuje. V záhlaví editoru je seznam skupin, kde má každá skupina název a indikátor počtu formulářových prvků, které obsahuje. Ve výchozím stavu existuje vždy alespoň jedna skupina, která je zároveň aktivní. Aktivní skupina je podtržena pruhem a editor zobrazuje prvky právě aktivní skupiny. Uživatel může skupiny přidávat, přejmenovávat a mazat pomocí ovládacích prvků umístěných v pravém horním rohu nad formulářovým editorem. Řazení skupin probíhá tažením názvu skupiny v záhlaví editoru na požadované místo.

Přidáním prvku ze sady do editoru se zobrazí náhled formulářového prvku. Náhled je rozdělen na dva řádky, kde první řádek obsahuje zprava ovládací prvky formulářového prvku, jeho název a indikátor možnosti *drag and drop* řazení. Druhý řádek zobrazuje samotný formulářový prvek. Pomocí ovládacích prvků může uživatel odstranit prvek z formuláře, vytvořit jeho kopii, přesunout ho do jiné skupiny a upravit jej. Tažením prvku pomocí indikátoru uživatel mění pořadí prvků v rámci aktivní skupiny. Při najetí kursoru myši na ovládací prvek se zobrazí text, který vystihuje akci po kliknutí. Tato funkcionalita byla přidána na základě uživatelského testování, kdy uživatelé často postupovali metodou pokus omyl při snaze upravit formulářový prvek.

Volbou úpravy prvku se zobrazí formulářový editor s možnostmi modifikace atributů a nastavení validačních pravidel. Zobrazeny jsou atributy a validátory uvedené v řídícím objektu daného formulářového prvku. Příklad rozdílných formulářových editorů je

na obrázku 10. Zajímavou funkcionalitou pro uživatele je náhled formulářového prvku přímo v editoru. Úpravy atributů jsou v reálném čase zobrazeny v náhledu a slouží pro kontrolu výsledku. Tím by měl klesnout počet chyb při úpravě atributů, které autor zaznamenal během uživatelského testování. Validace se nastavují na úrovni prvků pomocí tzv. validátorů. Validátor je pravidlo a jemu přiřazená chybová hláška. Validátorů může být definováno více a validace obecně jsou podrobně popsané v oddílu 4.7.2.



Obrázek 10 Editor formulářového prvku Text (vlevo) a Tree (vpravo)

V levém horním rohu nad formulářovým editorem je umístěn ovladač akcí importu a exportu tvořeného formulářového schématu. Uživatelé mohou mezi sebou sdílet rozestavěné formuláře nebo tvořit šablony pro rychlou inicializaci určitého formuláře. Po zvolení akce exportu formulářového schématu je aktuální proces uložen do *JSON* souboru a stáhnut do klientského počítače. Při importu je nejprve zkontrolována struktura nahrávaného *JSON* souboru a v pozitivním případě je podle něj inicializován *FormArchitect.*

4.7.2 FormRenderer

Komponenta *FormRenderer* vykresluje formulář, validuje zadané hodnoty a předává zadaná data k dalšímu zpracování. Na vstupu přijímá komponenta formulářové schéma vytvořené pomocí *FormArchitect* ve formě *JavaScript* objektu nebo *JSON*. Pokud předáme schéma jako *JSON* soubor, dojde k namapování názvů formulářových prvků na jejich odpovídající *UI* prvky. V opačném případě jsou reference na prvky součástí předávaného objektu a mapování není potřeba. Tímto způsobem je integrátor knihovny odstíněn od problémů předávání dat a kontroly jejich formátu. Může tedy sestavovat a vykreslovat formulář na oddělených místech aplikace, nebo napříč aplikacemi. Na vstupu je dále specifikovaná funkce, která parametrem přijímá validovaná data z formuláře a je vyvolána po uživatelském odeslání formuláře a úspěšné validaci zadaných hodnot. Je tedy zodpovědností integrátora knihovny, jak s daty dále naloží.

Knihovna *React.js* ve své dokumentaci uvádí 2 způsoby, jak pracovat s formulářovými vstupy. Kontrolovaný přístup umožňuje programátorovi řídit chování a překreslování formulářového prvku tím, že uchovává jeho hodnotu v paměti a v případě změny překreslí prvek s aktualizovanou hodnotou. Nekontrolovaný přístup nechává starost o hodnoty prvků na formuláři a programátor s formulářem pracuje pomocí odchytávaných událostí, stejně jako tomu je v prostředích mimo *React.js*. [31]

Přestože je jednodušší využít kontrolovaný přístup, časté překreslování vede ke zpomalení celé aplikace. Zejména v případech, kdy je na obrazovce vykresleno větší množství *UI* prvků. Z tohoto důvodu *FormRenderer* využívá nekontrolovaný přístup. Integrátor knihovny ovšem nemusí řešit často složitou práci s odchytáváním a zpracováním formulářových událostí a pracuje jen s odeslanými daty na výstupu, která jsou zároveň validovaná.

Data jsou automaticky validována pomocí tzv. validačního schématu knihovny *Yup.* Knihovna umožňuje testovat různé hodnoty pomocí předem nadefinovaných funkcí (validátorů) a vracet chybové hlášení, pokud test selže. Validační schéma vzniká *deklarativním stylem*, a proto může mít každý formulářový prvek více validačních pravidel a ke každému pravidlu specifickou chybovou hlášku. Obecně je schéma vytvořeno jako statický objekt programátorem. Pro potřeby *RFA* musel být tento přístup upraven tak, aby se schéma vytvořilo dynamicky na základě definic z předloženého formulářového schématu. Vytvořené řešení – před prvním vykreslením komponenty *FormRenderer* – programově sestaví validační schéma podle

definovaného validačního typu a validátorů každého prvku z formulářového schématu. Díky tomu, že je validační schéma tvořeno programově, vznikl systém validace, který je dostatečně robustní, flexibilní, jednoduše rozšiřitelný a využitelný při zobrazení jakéhokoliv formuláře bez nutnosti další vývojářské práce. V porovnání s obecnými přístupy k validacím a testovanými knihovnami z kapitoly 3 je tento systém unikátní a autor práce ho považuje, kromě samotné knihovny *RFA,* za největší příspěvek do open-source, který tato práce přináší.

Výstup dat z komponenty *FormRenderer* je ve formě *JavaScript* objektu, kde jsou názvy formulářových prvků a jejich hodnoty uvedeny jako páry klíč-hodnota. Příklad výstupu jednoduchého formuláře je na obrázku 11. V případě kolize názvu více prvků je vygenerován unikátní klíč skládající se z názvu prvku a *UUID* dle konvence pojmenování *kebab-case*.

Obrázek 11 Výstup dat z komponenty FormRenderer

4.7.3 Tree

Knihovna *RFA* v rámci nabízených komponent poskytuje možnost práce se stromově strukturovanými daty a tím zdůrazňuje své použití v prostředích, kde se taková data obecně vyskytují, tedy například v přírodních vědách. Řešený je celý životní cyklus takových dat, od jejich definice, přes vyplnění uživatelem, až po jejich interpretaci. Tento přístup je rozdílný oproti ostatním typům dat, kde *RFA* řeší pouze definici, zobrazení formuláře a interpretaci výsledků nechává na integrátorovi knihovny, dle jeho specifických potřeb. Zobrazení stromově strukturovaných dat není jednoduchý úkol, a proto si *RFA* bere za cíl zpřístupnit celý životní cyklus těchto dat včetně jejich interpretace a další následné práce. V rámci této podkapitoly bude přiblížena implementace celého cyklu napříč komponentami.

Pomocí komponenty *FormArchitect* probíhá přidání prvku stromu (*Tree*) do sestavovaného formuláře a definice struktury uzlu stromu jako je ukázáno na obrázku 10. Uzel stromu

se skládá z názvu, obsahu a potomků. Uživatel může specifikovat podobu označení uzlu a šablonu jeho obsahu. Obsah uzlu se skládá ze záznamů typu klíč-hodnota, kde uživatel může přidávat záznamy se specifikovaným klíčem a předvyplnit jejich hodnoty. Volitelně může také zpřístupnit možnost cizího přidání záznamu, čímž povolí možnost uživateli, který bude vyplňovat formulář se stromem, přidat uzlům vlastní záznamy, kromě již definovaných.

Komponenta *FormRenderer* vykreslí formulář se stromem dle formulářového schématu. Uživatel vidí první uzel – kořen stromu – se specifikovanou strukturou a postupným vyplňováním a přidáváním potomků tvoří výsledný strom. Navíc má možnost nechtěné uzly mazat a přebytečné větve skrývat.

O interpretaci dat se stará komponenta *Tree*. Na základě předložených stromově strukturovaných dat na vstup, zobrazuje výsledný strom. Uživatel může procházet strom a stejně jako při práci s předchozími komponentami, skrývat a odhalovat jednotlivé uzly. Pro přehlednější procházení je vždy vidět cesta k právě zobrazovanému uzlu s názvy jeho předchůdců. Po kliknutí na název uzlu v cestě se pohled stromu přenese na požadovaný uzel. Uživatel tak může prozkoumat hluboce zanořené uzly a zároveň vidět jejich původ, ke kterému se může kdykoliv jednoduše vrátit. Kromě zobrazení a procházení nabízí komponenta navíc funkci hledání ve stromě a výběru dat. Prohledávání probíhá podle hledaného výrazu, který se porovná s názvem a obsahem jednotlivých uzlů a případně zobrazí shodující se uzly. Po kliknutí na uzel jsou předána jeho data funkci, která je definovaná na vstupu komponenty *Tree* a následně vyvolána.

Přestože je zamýšleným použitím využít celý životní cyklus stromově strukturovaných dat tak, jak je popsáno výše, je možné komponentu *Tree* využít samostatně. V takovém případě integrátor knihovny ocení malou velikost komponenty a minimum závislostí, které obsahuje. K malé velikosti komponenty přispívá použitá rekurze, která se vhledem k povaze dat nabízí, a stylování provedené čistě pomocí *CSS* namísto použití balíčků, které vykreslují strom pomocí *SVG* prvků jako například *d3.js*[28] nebo *TreeModel.js*[29].

[28] https://d3js.org/

[29] http://jnuno.com/tree-model-js/

4.8 Dokumentace

Dokumentace knihovny je psaná pomocí značkovacího jazyku *kramdown,* který je nadstavbou známějšího značkovacího jazyku *Markdown*. Pomocí nástroje *Jekyll* jsou *kramdown* soubory přetransformované do statických *HTML* stránek a ty jsou hostované na *GitHub Pages*. *GitHub Pages* je platforma pro hostování statických webových stránek přímo z repositáře na *github.com*. Dokumentace projektu je tak propojena s jeho zdrojovými kódy a stejně jako kód je i verzována. Výhodou je, že *github.com* poskytuje tuto službu zdarma a statické *HTML* stránky jsou načítány velice rychle. Dokumentace je psaná v anglickém jazyce.

Knihovna obsahuje vstupní dokument, který se zobrazí při návštěvě repositáře v registru *npm* nebo na *github.com*, což jsou hlavní zdroje, ze kterých se může uživatel o knihovně dozvědět. Vstupní dokument obsahuje jen nejzákladnější informace o knihovně. Popsán je její účel, způsob instalace a příklad základního užití. Dále obsahuje odkaz na dokumentaci, který uživatele přesměruje na statické stránky.

Úvodní strana dokumentace obsahuje informace o knihovně, motivaci, která vedla k jejímu vytvoření a výpis důvodů proč knihovnu použít. Dále představuje a odkazuje na existenci komponent *FormArchitect, FormRenderer* a *Tree*. Poslední částí úvodní stránky je interaktivní příklad, kde si může uživatel naživo vyzkoušet sestavení, zobrazení a odeslání formuláře. Příklad je vložen jako *iframe* s virtualizovaným prostředím pomocí *codesandox.io*[30]. To umožňuje uživateli zobrazit výsledek, stejně jako zdrojový kód, a navíc provádět a sledovat změny.

Strana „First steps“ - neboli první kroky – popisuje způsob instalace a příklad kódu se základním použitím knihovny. Tato strana se shoduje se vstupním dokumentem popsaným výše. Strana „Components“ – neboli komponenty – obsahuje seznam knihovnou nabízených komponent, popis jejich API a příklady základního použití. Tím, že je knihovna napsaná pomocí jazyka *TypeScript* jsou uvedeny i typy vstupních a výstupních parametrů každé komponenty.

Strana „Tree“ – neboli strom – zobrazuje informace nejen o stejně pojmenované komponentě, ale o celkovém zpracování životního cyklu stromově strukturovaných dat, který knihovna

[30] https://codesandbox.io/

nabízí. Popsány jsou kroky v jednotlivých komponentách vedoucí až k interpretaci dat pomocí komponenty *Tree* podobně, jako je tomu v oddílu 4.7.3. Navíc strana obsahuje obrazové ukázky jednotlivých kroků a animaci procházení rozsáhlého stromu.

Poslední strana dokumentace obsahuje složitější příklady užití knihovny, jako je například přizpůsobení barev podle prostředí, do kterého je knihovna integrována nebo rozšíření knihovny vlastním formulářovým prvkem. Příklady kromě popisu a ukázky kódu obsahují i odkazy na *codesandbox.io,* kde je možné vidět užití v reálném projektu, stejně jako je tomu na úvodní straně dokumentace.

4.9 Testování

Knihovna je testována na úrovni svých funkčních celků. Každá komponenta obsahuje svou sadu testů, které kontrolují její funkčnost. Jsou zvoleny dva typy testovacích sad. Jednotkové testy kontrolují funkčnost kódu po jeho celcích a v kódu jsou implementovány pomocí knihovny *Jest*[31]. End-to-end testování je implementováno pomocí testovacího nástroje *Cypress*[32], který umožňuje nadefinovat akce ve virtualizovaném prohlížeči, kde běží testovaná aplikace, a tím simulovat chování reálného uživatele.

Protože je knihovna publikovaná jako open-source projekt, do kterého může kdokoliv přispět, je důležité mít definovaný automatizovaný proces kontroly přijímaného kódu. Tento proces se skládá ze zřetězených akcí, po jejichž provedení dojde k přijmutí nebo zamítnutí přidávaného příspěvku. Po provedení požadavku na změnu nebo přidání nového kódu jsou spuštěny testy a na základě výsledku je akce přerušena nebo pokračuje na další. Knihovna *RFA* tento proces implementuje pomocí *GitHub Actions*[33]*,* což je implantace zřetězených akcí nad repositářem na serveru *github.com.*

4.10 Komparace s konkurenty

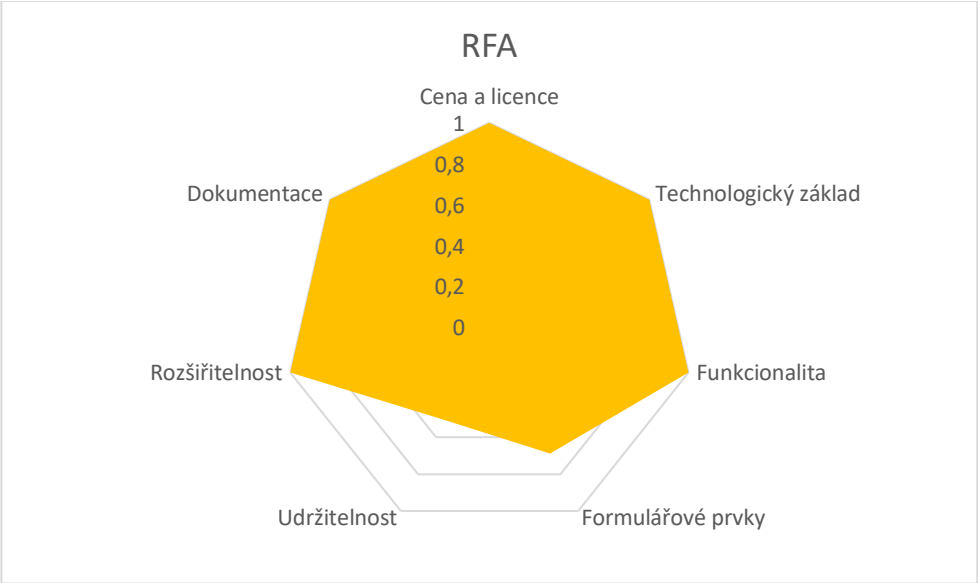
Knihovna je podrobena stejné analýze z vývojářského a uživatelského hlediska jako jsou konkurenční řešení v kapitole 3. Z vývojářského hlediska obdržela knihovna 60,4 bodu z 70 možných, tedy 86,3 %, a je proto hodnocena nejlépe v porovnání se svými konkurenty.

[31] Framework pro testování JavaScript aplikací. Dostupné z https://jestjs.io.

[32] Framework pro tvorbu end-to-end testů. Dostupné z https://www.cypress.io.

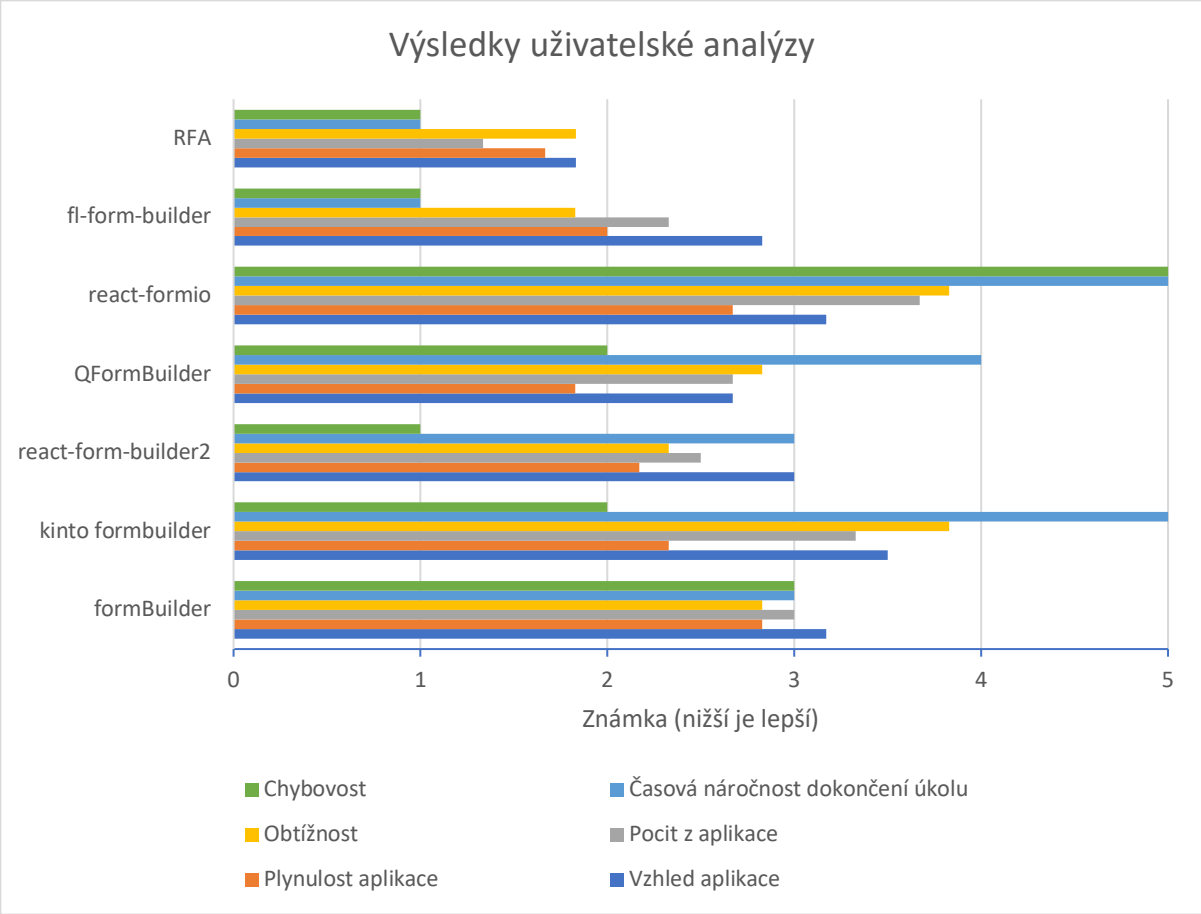
[33] https://github.com/features/actions

Knihovna ztrácí body v oblasti udržitelnosti a počtu nabízených formulářových prvků. Obě tyto oblasti souvisí se stářím knihovny a očekává se, že hodnocení poroste postupem času a doplňováním nových funkcí. Výsledek vývojářské analýzy *RFA* je zobrazen na grafu 4.



Graf 4 Výsledek vývojářské analýzy RFA

Uživatelské testování bylo uskutečněno na vzorku 6 lidí, s odstupem 6 měsíců od testování pro analýzu z kapitoly 3. Z toho 4 účastníci se zúčastnili i prvního kola testování a probíhalo dle stejného scénáře (Příloha A) jako testování konkurentů. Výsledky jsou zobrazeny na grafu 5. Uživatelé hodnotí práci s knihovnou pozitivně a ze všech testovaných řešení dosahuje *RFA* nejlepších výsledků. Průměrně trvalo dokončení zadaného úkolu 3:01 minuty s 1,2 chybou na uživatele. Dle těchto kritérií dochází použitím knihovny ke znatelnému posunu oproti konkurentům.



Graf 5 Výsledek uživatelského testování RFA v porovnání s konkurenty

Výsledek komparace označuje knihovnu *RFA* za nejvhodnější řešení. Z technického hlediska stále existují slabá místa, která ovšem počítají s budoucím rozvojem, který je u *open-source* projektů očekáván. Z *UX* hlediska se zdá být knihovna uživatelsky přívětivá, ovšem i zde autor identifikoval místa, která by stála za vylepšení. Získané poznatky budou využity při dalším rozvoji aplikace.

5 Závěr

Tato práce se zabývala problematikou definování a vizualizace dynamických webových formulářů. Konkrétněji se pak zaměřovala na problematiku vytváření datových schémat prostřednictvím webového uživatelského prostředí a na práci se stromově strukturovanými daty. Hlavním cílem práce pak byla tvorba vlastní knihovny pro definici datových struktur.

V tomto kontextu lze konstatovat, že práce naplnila svůj hlavní cíl. Vznikla knihovna *RFA*, která umožňuje uživatelskou definici různých datových struktur ve webovém prostředí pomocí formuláře. Kromě základních formulářových prvků poskytuje knihovna navíc prvek pro práci se stromově strukturovanými daty, čímž zdůrazňuje své použití v přírodních vědách. Kvůli své univerzálnosti je však vytvořené řešení využitelné i v jiných oborech a prostředích. Knihovna poskytuje komponenty pro uživatelské sestavení formuláře, jeho vykreslení a interpretaci stromově strukturovaných dat. Je implementovaná takovým způsobem, aby byla přívětivá i pro běžného uživatele. Je volně dostupná jako *open-source* balíček v registru *npm,* se zdrojovými kódy v repositáři na *github.com,* kde je také dostupná její rozsáhlá dokumentace.

Potřeba vytvoření knihovny vychází z rešerše aktuálního stavu publikace a zpracování dat v přírodních vědách. Situace bio-databází dostupných na internetu je v přístupu ke zpracování heterogenních dat shodná. Pracuje se pouze se statickou strukturou dat a data, která nelze na takovou strukturu namapovat, jsou zahozena. Rozšíření nebo definice nové struktury je v režii vývojáře. S rychlostí přibývání nových dat je ovšem tento přístup časově a finančně neudržitelný.

Analyzována jsou také současná řešení, která umožňují sestavení formuláře ve webovém prostředí. Průzkum probíhá z technického a uživatelského pohledu. Technický průzkum definuje ideální řešení, kterému se ovšem žádné z analyzovaných knihoven dostatečně neblíží. Uživatelé hodnotí zkoumané aplikace spíše průměrně a ty, které dosáhly nejvyššího výsledku v technické analýze, řadí mezi nejhůře hodnocené. Žádné z existujících řešení navíc nenabízí možnost práce se stromově strukturovanými daty, a proto limitují možnost svého použití ve vědeckých prostředích.

Z provedených rešerší a analýz vyplynula potřeba vlastního řešení (tedy nejen například rozšíření stávající knihovny), které by disponovalo požadovanou funkcionalitou, a navíc bylo využitelné běžným uživatelem, tedy nejen vývojářem. Takové řešení je následně navrženo a

implementováno, jak je popsáno v textu práce. Vyzdvihnout lze především funkcionalitu shlukování formulářových prvků do skupin, nastavení validačních pravidel, dynamickou úpravu atributů, export a import schématu tvořeného formuláře, možnost definice vlastního formulářového prvku a přizpůsobení vzhledu nabízených komponent.

Knihovna *RFA* při technické komparaci poráží své konkurenty a uživatelské testování odhaluje, že práce s knihovnou je rychlejší a méně chybová než práce s konkurenčními knihovnami při plnění stejného úkolu. Dle hodnocení účastníků testování je knihovna uživatelsky přívětivá. Během prvního měsíce od publikace má knihovna již přes 250 stáhnutí dle registru *npm* a má reálné využití v projektu univerzální nálezové databáze (*UniCatDB*[34]) v rámci organizace *ELIXIR*.

Knihovna je dostupná ve svém repositáři na adrese *https://github.com/Kamony/rfa,* nebo v registru *npm* pod odkazem *https://www.npmjs.com/package/rfa.* Dokumentace je samostatně dostupná na *https://kamony.github.io/rfa.*

[34] https://www.unicatdb.org/

6 Literatura

[1] WU, Hongyan a Atsuko YAMAGUCHI. *Semantic Web technologies for the big data in life sciences*. 8. AUG 2014, s. -201, 192 s. ISSN 18817815. Dostupné z: doi:10.5582/bst.2014.01048

[2] SZALAY, Alexander a Jim GRAY. 2020 Computing: Science in an exponential world: Science in an exponential world. *Nature*. 2006/04/01, **440**, 413-4. Dostupné z: doi:10.1038/440413a

[3] PAGE, Lawrence M., Bruce J. MACFADDEN, Jose A. FORTES, Pamela S. SOLTIS a Greg RICCARDI. Digitization of Biodiversity Collections Reveals Biggest Data on Biodiversity. *BioScience*. 2015, **65**(9), 841-842. ISSN 0006-3568. Dostupné z: doi:10.1093/biosci/biv104

[4] ELIXIR. ELIXIR Scientific Programme 2019-2023 [online; version 1; not peer reviewed]. *F1000Research* 2019, **8**(ELIXIR):1570 [cit. 2020-01-24]. Dostupné z: https://doi.org/10.7490/f1000research.1117352.1

[5] SIMPSON, Dana. *Use of Big Data: Benefits, Risks, and Differential Pricing Issues: Business Economics in a Rapidly-changing World*. New York: Nova Science Publishers, 2016. ISBN 9781634853194.

[6] SAXENA, Shilpi a Saurabh GUPTA. *Practical Real-Time Data Processing and Analytics*. Birmingham, UK: Packt Publishing, 2017 [cit. 2020-03-01]. ISBN 9781787281202.

[7] BUYYA, Rajkumar, Rodrigo N. CALHEIROS a Amir Vahid DASTJERDI, ed. *Big data: Principles and paradigms*. Amsterdam: Elsevier, 2016. ISBN 978-0-12-809346-7.

[8] EL ARASS, Mohammed a Nissrine SOUISSI. Data Lifecycle: From Big Data to Smart Data. *5THEDITION INTERNATIONAL IEEE CONGRESS on INFORMATION SCIENCE and TECHNOLOGY* [online]. 2018, 23.10.2018 [cit. 2020-03-04]. Dostupné z: https://www.researchgate.net/publication/328769944_Data_Lifecycle_From_Big_Data_to_Smart_Data

[9] WIESE, Lena. *Advanced data management: for SQL, NoSQL, cloud and distributed databases*. Boston: De Gruyter, Oldenbourg, [2015]. ISBN 9783110441406.

[10] Tutorial: Get Started with Entity Framework 6 Code First using MVC 5. *Microsoft* [online]. c2020, 22.01.2019 [cit. 2020-03-04]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application

[11] Django Tutorial Part 9: Working with forms. *MDN web docs: moz://a* [online]. c2005-2020 [cit. 2020-03-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Forms

[12] Tour of Heroes app and tutorial. *Angular* [online]. c2010-2020 [cit. 2020-03-04]. Dostupné z: https://angular.io/tutorial

[13] *GBIF: the Global Biodiversity Information Facility* [online]. Copenhagen, 2020 [cit. 2020-11-19]. Dostupné z: https://www.gbif.org/

[14] BALL-DAMEROW, Joan E., Laura BRENSKELLE, Narayani BARVE, et al. Research applications of primary biodiversity databases in the digital age. *PLOS ONE*. 2019, **14**(9). ISSN 1932-6203. Dostupné z: doi:10.1371/journal.pone.0215794

[15] RATNASINGHAM, SUJEEVAN a PAUL D. N. HEBERT. Bold: The Barcode of Life Data System (http://www.barcodinglife.org): The Barcode of Life Data System (http. *Molecular Ecology Notes*. John Wiley, 2007/05/01, **7**(3), 355-364. ISSN 1471-8278. Dostupné z: https://doi.org/10.1111/j.1471-8286.2007.01678.x

[16] *OBIS* [online]. Oostende, 2020 [cit. 2020-11-20]. Dostupné z: https://www.obis.org/

[17] *AVH* [online]. Canberra, 2020 [cit. 2020-11-20]. Dostupné z: https://avh.chah.org.au/

[18] BEAMAN, Reed a Nico CELLINESE. Mass digitization of scientific collections: New opportunities to transform the use of biological specimens and underwrite biodiversity science: New opportunities to transform the use of biological specimens and underwrite biodiversity science. *ZooKeys*. Pensoft Publishers, 2012, **209**(-), 7-17. ISSN 1313-2989. Dostupné také z: https://doi.org/10.3897/zookeys.209.3313

[19] The State of the OCTOVERSE. *GitHub.com* [online]. c2019 [cit. 2020-02-13]. Dostupné z: https://octoverse.github.com/

[20] HEITKOTTER, Henning, Tim A. MAJCHRZAK, Benjamin RULAND a Till WEBER. *Evaluating Frameworks for Creating Mobile Web Apps* [online]. 2013, 209-221 [cit. 2020-02-11]. Dostupné z: https://www.researchgate.net/publication/259852798_Evaluation_of_cross-platform_frameworks_for_mobile_applications

[21] OLARU, Alexander. Selection Criteria for Javascript Frameworks. *InfoQ.com* [online]. c2006-2019 [cit. 2020-02-11]. Dostupné z: https://www.infoq.com/news/2007/12/choosing-javascript-frameworks/

[22] KRUG, Steve. Don't make me think, revisited: a common sense approach to web usability. 2014. San Francisco: New Riders, 2014. ISBN 9780321965516.

[23] UNGER, Russ a Carolyn CHANDLER. *A project guide to UX design: for user experience designers in the field or in the making*. Berkeley, CA: New Riders, c2009. ISBN 9780321607379.

[24] MURPHY, Christopher. A Comprehensive Guide To User Testing. *Smashing Magazine* [online]. Freiburg: Smashing Media, 07.03.2018 [cit. 2020-02-25]. Dostupné z: https://www.smashingmagazine.com/2018/03/guide-user-testing/

[25] FIALA, Petr, Miroslav MAŇAS a Josef JABLONSKÝ. *Vícekriteriální rozhodování*. Praha: Vysoká škola ekonomická, 1994. ISBN 80-7079-748-7.

[26] TURNER, Carl W., James R. LEWIS a Jakob NIELSEN. Determining Usability Test Sample Size. *International Encyclopedia of Ergonomics and Human Factors* [online]. 2006, (3) [cit. 2020-02-25]. Dostupné z: https://www.researchgate.net/publication/242156700_Determining_Usability_Test_Sample_Size

[27] NIELSEN, Jakob. How Many Test Users in a Usability Study? *Nielsen Normal Group* [online]. c1998-2020, 03.05.2012 [cit. 2020-02-26]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/

[28] BROOKSHEAR, J.G. *Informatika*. Albatros Media, 2015. ISBN 9788025142677.

[29] Kinto 13.7 documentation. *Kinto* [online]. Mozilla Services, c2015-2020 [cit. 2020-02-14]. Dostupné z: https://kinto.readthedocs.io/en/latest/index.html

[30] A Form and Data Management Platform for Progressive Web Applications. *Form.io* [online]. Dallas, TX: Form.io LLC., c2020 [cit. 2020-02-17]. Dostupné z: https://www.form.io/

[31] Controlled vs. Uncontrolled Components. *React* [online]. Menlo Park (California): Facebook Open Source, c2020 [cit. 2020-11-30]. Dostupné z: https://reactjs.org/docs/glossary.html#controlled-vs-uncontrolled-components

7 Seznam obrázků

Obrázek 1 FormBuilder 18
Obrázek 2 Kinto FormBuilder 19
Obrázek 3 React Form Builder 21
Obrázek 4 QFormBuilder 22
Obrázek 5 react-formio 24
Obrázek 6 fl-form-builder 25
Obrázek 7 Diagram případů užití RFA 43
Obrázek 8 Struktura RFA 45
Obrázek 9 Popis částí komponenty FormArchitect 46
Obrázek 10 Editor formulářového prvku Text (vlevo) a Tree (vpravo) 48
Obrázek 11 Výstup dat z komponenty FormRenderer 50

8 Seznam grafů

Graf 1 Rozdělení vah vývojářských kritérií....11
Graf 2 Výsledky vývojářského hodnocení knihoven....30
Graf 3 Výsledky uživatelského hodnocení knihoven....31
Graf 4 Výsledek vývojářské analýzy RFA....54
Graf 5 Výsledek uživatelského testování RFA v porovnání s konkurenty....55

9 Seznam tabulek

Tabulka 1 Nejcitovanější bio-databáze 4
Tabulka 2 Výsledky rešerše nástrojů pro tvorbu dynamických formulářů 9
Tabulka 3 Vývojářská hodnotící kritéria 12
Tabulka 4 Uživatelská hodnotící kritéria 16
Tabulka 5 Výsledek analýzy dle vývojářských kritérií 27
Tabulka 6 Výsledek analýzy uživatelského testování 32
Tabulka 7 Funkční požadavky 34
Tabulka 8 Nefunkční požadavky 35
Tabulka 9 Logický rámec projektu 36

Příloha A – Scénář uživatelského testování

Dobrý den, jmenuji se Jiří Hauser a v současné době píšu diplomovou práci na UAI PŘF JČU. Tématem mé práce je vytvoření nástroje pro tvorbu dynamických webových formulářů. Součástí je i analýza již existujících nástrojů, které se svojí funkcionalitou mé práci blíží. Jedním z faktorů, které zkoumám je uživatelská přívětivost jednotlivých řešení. Prosím Vás tedy, jako účastníka testování, o hodnocení 6 nástrojů pro tvorbu dynamických formulářů. Testování bude probíhat následujícím způsobem.

Zadám Vám jednoduchý úkol, který se pokusíte splnit v každém z testovaných nástrojů. Úkol je definován tak, že ho lze splnit ve všech testovaných nástrojích.

Já budu s Vámi po celou dobu testu a budu sledovat a zaznamenávat jeho průběh. Pamatujte prosím, že je testován nástroj a ne Vy. Nevadí tedy, pokud úkol nesplníte nebo se při jeho plnění zaseknete. Při plnění úkolu Vám nemohu pomáhat. Pokud budete mít během plnění úkolu otázky, jak pokračovat dál nebo používat daný nástroj, tak vězte, že Vám nemohu odpovědět a zkuste daný problém vyřešit vlastní pomocí.

Vždy po dokončení úkolu Vás poprosím o vyplnění krátkého dotazníku, který se zaměřuje na použitelnost a vzhled testovaného produktu. Dotazník obsahuje 4 otázky, které hodnotíte na stupnici 1 až 5, jako ve škole.

Úkol: ***Vytvořte objednávkový formulář pro zákazníky pizzerie. Formulář by měl obsahovat:***

- ***pole pro vyplnění adresy zákazníka***
- ***zaškrtávací políčka pro přidání cibule nebo slaniny navíc***
- ***výběrové pole s možnostmi doručení domů nebo vyzvednutí na prodejně***
- ***přepínací tlačítko s výběrem zasílání reklamních nabídek***