TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Analýza a optimalizace softwarových nástrojů test managementu pro Testcentrum elektroniky ve Škoda Auto

Diplomová práce

Studijní program:	N2612 – Elektrotechnika a informatika
Studijní obor:	1802T007 – Informační technologie

Autor práce:	**Bc. Vladimír Škopek**
Vedoucí práce:	Ing. Igor Kopetschke

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Analýza a optimalizace softwarových nástrojů test managementu pro Testcentrum elektroniky ve Škoda Auto

Jméno a příjmení: **Bc. Vladimír Škopek**
Osobní číslo: M17000139
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Proveďte rešerši aktuálně používaného softwaru v test managementu Škoda Auto.
2. Na základě rešerše zvolte funkce a případná vylepšení současných (roztříštěných) řešení.
3. Navrhněte vlastní webovou aplikaci pro potřeby test managementu včetně struktury pro uchovávání dat.
4. Implementujte navržené řešení za použití frameworku Django a PostgreSQL.
5. Aplikaci otestujte ve zkušebním provozu Škoda Auto a zhodnoťte dosažený výsledek.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	40 – 50 stran
Forma zpracování práce:	tištěná/elektronická

Seznam odborné literatury:

[1] SMITH, Gregory. PostgreSQL 9.0: high performance : accelerate your PostgreSQL system and avoid the common pitfalls that can slow it down. Birmingham: Packt Publishing, c2010. ISBN 978-1849510301.
[2] FORCIER, Paul. Python web development with Django. Boston, MA: Addison Wesley, 2009. Developer´s library. ISBN 9780132356138.
[3] LUTZ, Mark a David ASCHER. Learning Python. 2nd ed. Sebastopol, Calif.: O'Reilly, 2004. ISBN 0-596-00281-5.

Vedoucí práce:	Ing. Igor Kopetschke Ústav nových technologií a aplikované informatiky
Datum zadání práce:	18. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci 18. října 2018

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 29. 4. 2019

Podpis:

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce panu Ing. Igorovi Kopetschkemu za jeho odborné vedení a vstřícnost. Velký dík patří mému vedoucímu stáže ve Škoda Auto Ing. Liborovi Heřmanovi za užitečné rady a množství času, který mi při konzultacích v průběhu roku i přes velké pracovní vytížení věnoval.

Abstrakt

Práce se zabývá problematikou test managementu v automobilovém průmyslu. Na základě rešerše současných řešení vznikla nová test management aplikace pro Testcentrum elektroniky ve společnosti Škoda Auto. Serverová část aplikace je realizována ve webovém frameworku Django v jazyce Python, klientská část je napsána v jazyce JavaScript, pro uchovávání dat slouží databáze PostgreSQL. Nově vzniklá aplikace umožňuje zadávání požadavků na test, správu testovacích případů, manuální testování a zobrazování výsledků testů ve formě sloupcových a koláčových grafů. Předložené řešení využívá systém uživatelských účtů a oprávnění k oddělení přístupu do jednotlivých částí aplikace podle definovaných rolí. Součástí implementace je také uživatelská administrace databáze. Aplikace byla otestována ve zkušebním provozu Škoda Auto.

Klíčová slova

Webová aplikace, Django, Python, JavaScript, testování, test management, Škoda Auto

Abstract

The thesis deals with test management in automotive industry. Based on a review of current solutions a new test management application was developed for the Testcentre of Electronics of Škoda Auto. The server part of the application is implemented in the Django web framework in Python, the client part is written in JavaScript, the PostgreSQL database is used for data storage. The newly created application allows submitting test requests, managing test cases, manual testing and displaying test results in the form of bar and pie charts. The presented solution uses a system of user accounts and permissions to separate access to individual parts of the application according to defined roles. The implementation also includes user interface for the administration of the database. The application was tested via a trial run in Škoda Auto.

Keywords

Web application, Django, Python, JavaScript, testing, test management, Skoda Auto

Obsah

Seznam obrázků

Seznam zkratek a termínů

AJAX	Asynchronous JavaScript and XML
ALM	Application Lifecycle Management
CAN	Controller Area Network
CSS	Cascading Style Sheets
dTCM	Doors Test Case Manager
FO	Function owner
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
KPM	Konzern Problem Management
ORM	Object-relational mapper
PDF	Portable Document Format
RDBMS	Relational database management systém
RQM	Rational Quality Manager
TErU	Test Ergebnisse Übersicht
TOV	Test Objekt Verantwortlich
URL	Uniform Resource Locator
WSGI	Web Server Gateway Interface
XLS	eXceL Spreadsheet
XML	eXtensible Markup Language

Úvod

Nástup takzvaných chytrých zařízení a digitalizace do každodenního života člověka se v posledním desetiletí promítá i do automobilového průmyslu. S rostoucím počtem služeb, které moderní vozidla nabízí, se zvyšuje i složitost vývoje a následného testování. V roce 1968 Volkswagen představil ve spolupráci s firmou Bosch první počítačem řízené vstřikování paliva [1]. Technicky nejvyspělejší vozidla dneška disponují desítkami až stovkou řídicích jednotek.

Elektronika nese odpovědnost za kontrolu nad téměř všemi podstatnými ději v automobilu od stahování oken, přes multimédia až po automatické řazení. Neodhalená chyba v některém ze systémů může mít i fatální dopad na bezpečí pasažérů vozidla. Testování je nezbytnou součástí vývoje automobilů dnes i v blízké budoucnosti, s příchodem autonomních vozů se dá očekávat prudký nárůst scénářů, které bude nutné testovat, a proto je velmi důležité se daným tématem dále zabývat a podporovat rozvoj tohoto odvětví.

Při současných vysokých nárocích na rychlost a důslednost testování musí nejen management oddělení využívat softwarové nástroje, které usnadňují organizaci a běh takto komplexního procesu, zejména v oblasti plánování, vizuální prezentace výsledků a uchovávání dat. Cílem této diplomové práce je navrhnout a vyvinout vlastní software pro potřeby test managementu v oddělení Testcentra elektroniky ve Škoda Auto, a to vzhledem ke vzrůstající komplexitě elektronických systému vozidel, kde se současně využívané nástroje jeví jako nedostatečné.

Součástí práce je také analýza stávajících test management nástrojů. Získané informace budou stěžejní pro následnou implementaci nového systému. Další částí práce je rešerše existujících aplikací na trhu zaměřených na proces testování.

Diplomová práce je obsahově rozdělena na dvě části. První tři kapitoly jsou věnovány teoretickému úvodu a rešerši softwarových test management nástrojů ve Škoda Auto i mimo firmu. Čtvrtá až šestá kapitola obsahuje popis vývoje vlastního řešení. Jednotlivé kapitoly se zabývají návrhem řešení, jeho implementací a získanou zpětnou vazbou během zkušebního provozu. V závěru jsou shrnuty výsledky práce a nastíněny možné směry dalšího vývoje.

1 Úvod do problematiky

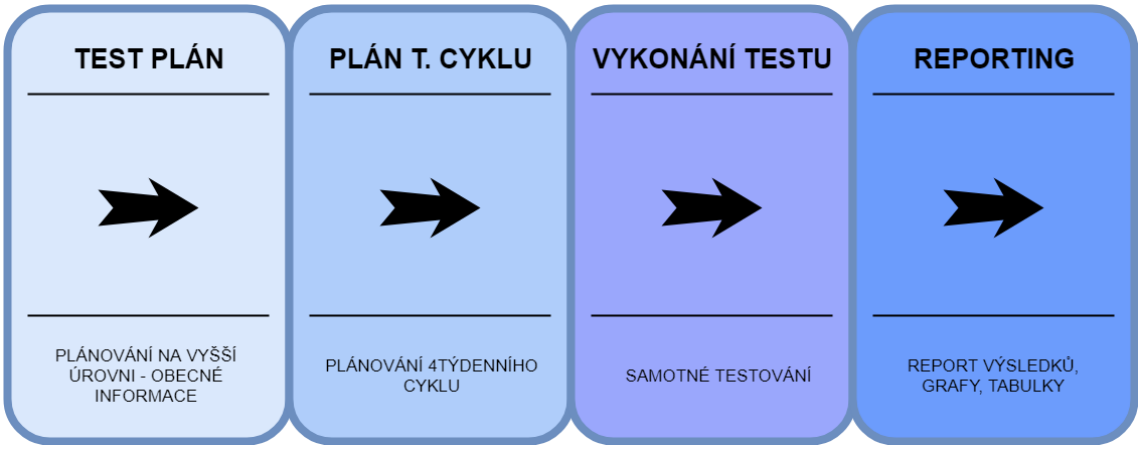
Diplomová práce se zabývá problematikou test managementu v oboru testování elektroniky v automobilovém průmyslu z hlediska softwaru. Pro zasazení do kontextu a lepší porozumění tématu, které úzce souvisí s nově vzniklou aplikací, jsou v této kapitole uvedeny základní informace a pojmy spojené s pracovištěm, na kterém vzniklo zadání práce.

1.1 Pracoviště

Oddělení Testcentrum elektroniky je v rámci společnosti Škoda Auto zodpovědné za provádění zkoušek a systémovou podporu vývoje elektroniky vozu. Konkrétně se zabývá plánováním integračních testů, laboratorním testováním řídicích jednotek zapojených v palubní síti vozu, testováním elektromagnetické kompatibility komponent, zkouškami CAN komunikace a dalšími činnostmi.

1.2 Testovací proces

Testovací proces je možné zjednodušeně rozdělit na čtyři části (viz Obrázek 1). Obecný test plán popisuje, jaký hardware a software se bude testovat a na jakém testovacím místě se bude testovat, a udává časový harmonogram testů. Testovací cyklus je sekvence úkonů, které musí proběhnout od začátku do konce testování (plánování testů, jejich analýza a příprava, provedení a vyhodnocení a sledování chyb). Po plánování následuje vykonání testů a reporting výsledků.



Obrázek 1: Testovací proces

1.3 Terminologie

S testováním v Testcentru elektroniky jsou spojené specifické termíny, které jsou dále v práci používány. V této podkapitole jsou stručně vysvětleny ty, které se v textu práce opakují.

1.3.1 Projekt

V souvislosti se Škoda Auto je projektem myšlen konkrétní model vozu, který automobilka produkuje (např. Škoda Octavia). Jednotlivé projekty se z hlediska testovaní elektroniky liší v osazení řídicích jednotek i senzorů.

1.3.2 Infotainment

Infotainment je termín automobilového průmyslu odkazující na systémy vozidel, které kombinují zábavu a poskytování informací řidičům i cestujícím. Pro poskytování těchto služeb využívají audio / video rozhraní, dotykové obrazovky a další zařízení [2].

1.3.3 Testovací případ

Testovací případ je dokument, který se skládá ze sady podmínek a akcí, které jsou prováděny na testovaném systému, aby se ověřila očekávaná funkčnost prvku [3]. Testovací případ obvykle obsahuje cíl, podmínky, které musí být splněny před provedením testu, jednotlivé kroky testu a očekávaný výsledek.

1.3.4 Testovací specifikace

Testovací specifikace je soubor testovacích případů vztahujících se k určité funkci, řídicí jednotce a testovacímu místu. Testovací specifikace jsou ve Škoda Auto generovány z aplikace IBM Rational Doors ve formátu DXML.

1.3.5 Funkční témata a podtémata

Testovací případy jsou v testovací specifikaci rozděleny do menších celků, tzv. funkčních témat a podtémat. Například jedním z témat testovací specifikace funkce *Audio Management* centrální řídicí jednotky infotainmentu je *Mute-tests*, toto téma pak obsahuje podtémata *Mute, Demute* a další.

2 Test management nástroje ve Škoda Auto

V současnosti Testcentrum elektroniky nepoužívá jeden ucelený testovací nástroj, který by pokryl celý proces testování. V různých fázích testování se používají menší aplikace se specifickým zaměřením na jeden daný úkol.

Tato kapitola popisuje současné softwarové vybavení spojené s procesem testování v Testcentru elektroniky. Po obecném popisu následuje podrobnější přehled ke každému z nástrojů ve vlastní podkapitole. Součástí popisu je upozornění na nedostatky některých nástrojů a návrh na jejich řešení.

Pro zadávání požadavků na test se používá aplikace Teamweb. Testovací specifikace se exportují z aplikace IBM Doors a k jejich čtení a provedení testů slouží aplikace dTCM. Chyby nalezené v průběhu testování se zadávají do aplikace KPM. Výsledky testů se zobrazují pomocí aplikace TErU.

Kromě zmíněných nástrojů se používá také tradiční kancelářská sada Microsoft Office a několik úzce specializovaných nástrojů. Jeden z nich porovnává starou a novou verzi testovací specifikace a zobrazuje rozdíly mezi nimi. Dále se používá nástroj pro vymezení testovací specifikace požadované funkce podle konkrétního požadavku zadavatele testu. Podrobnější analýzou těchto nástrojů se v práci nebudu zabývat. Test management aplikace, která vznikla v rámci této práce, předchází problémům, kvůli kterým je třeba tyto nástroje používat.

Jelikož většina těchto nástrojů vznikala nezávisle na sobě v různou dobu, není jejich provázání vždy ideální. Například výstupní data z jedné aplikace je nutné před použitím na vstupu do následující aplikace upravovat pomocí dalšího nástroje. Výrazně tak narůstá režie manipulace s daty. Dalším problémem je způsob, jakým nástroje k datům přistupují a jak je ukládají. Testovací specifikace a výsledky testů jsou uloženy na disku v textovém souboru ve formátu DXML. Při používání tohoto formátu může dojít k nežádoucím změnám ve výsledcích testů neoprávněnou osobou. Další nevýhodou tohoto přístupu je vyšší náročnost na udržování historie testování a také vyšší nároky na kapacitu disku.

Množství a roztříštěnost používaných nástrojů přináší několik nevýhod, z tohoto důvodu vzniklo zadání pro tuto diplomovou práci. Cílem je na základě analýzy současného stavu softwarového vybavení navrhnout vhodnější ucelený systém, který eliminuje nadbyteč-

nou režii při práci s daty a v případě potřeby vylepší funkce a vlastnosti vybraných nástrojů.

2.1 Teamweb

Teamweb je webová aplikace, která se ve Škoda Auto používá pro vytváření jednoduchých webových aplikací bez psaní kódu (uživatel může v omezené míře zasahovat do HTML kódu). Vytváření probíhá přímo v prohlížeči přidáváním předem připravených komponent z aplikační lišty Teamwebu do HTML stránky.

2.1.1 Teamweb Testcentrum Infotainment

V Testcentru elektroniky slouží Teamweb především pro zadávání požadavků na testování infotainmentu. Požadavek na test vznáší vlastník funkce, který určuje, jaké testovací případy mají být otestovány ve zvoleném kalendářním týdnu.

Uživatel Teamwebu může mít jednu ze tří rolí – designér, přispěvatel, čtenář. Designérem je uživatel, který může měnit podobu a obsah webu (vývojář s veškerými právy). Přispěvatel smí zadávat a později upravovat požadavky na testy a číst veškerý obsah webu (zaměstnanci, kteří mají zodpovědnost za danou funkci, jejíž test požadují). Čtenáři mohou pouze číst obsah webu.

Kromě formuláře pro zadání požadavku na test obsahuje Teamweb statické HTML stránky s důležitými informacemi o oddělení Testcentrum Infotainment. Je zde návod pro testery na zadávání chyb do systému KPM, informace o fyzických hardwarových testovacích nástrojích (snímač obrazovky, inteligentní pojistky) nebo například zaměstnanecká hierarchie oddělení.

2.1.2 Nevýhody Teamwebu

Jednoduchost vytváření webu v aplikaci Teamweb s sebou nese také nevýhody. Ta je navržena tak, aby byl uživatel schopen realizovat vlastní webové stránky bez jakýchkoliv předchozích zkušeností s jejich tvorbou. Při tvorbě stránek je uživatel odkázán pouze na nabídku komponent v horní aplikační liště, možnosti customizace a realizace složitější funkcionality jsou tímto velmi omezené. Teamweb je především vhodný pro vytváření informačních statických HTML stránek. Implementace vlastní dynamické práce s daty není možná.

Nejdůležitější funkce Teamwebu Testcentrum Infotainment, zadávání požadavků na test řídicích jednotek, trpí kvůli tvůrčím omezením platformy Teamweb mnoha nedostatky. Pro vysvětlení těchto nedostatků je v odstavci níže nejdříve popsán postup zadání požadavku na test.

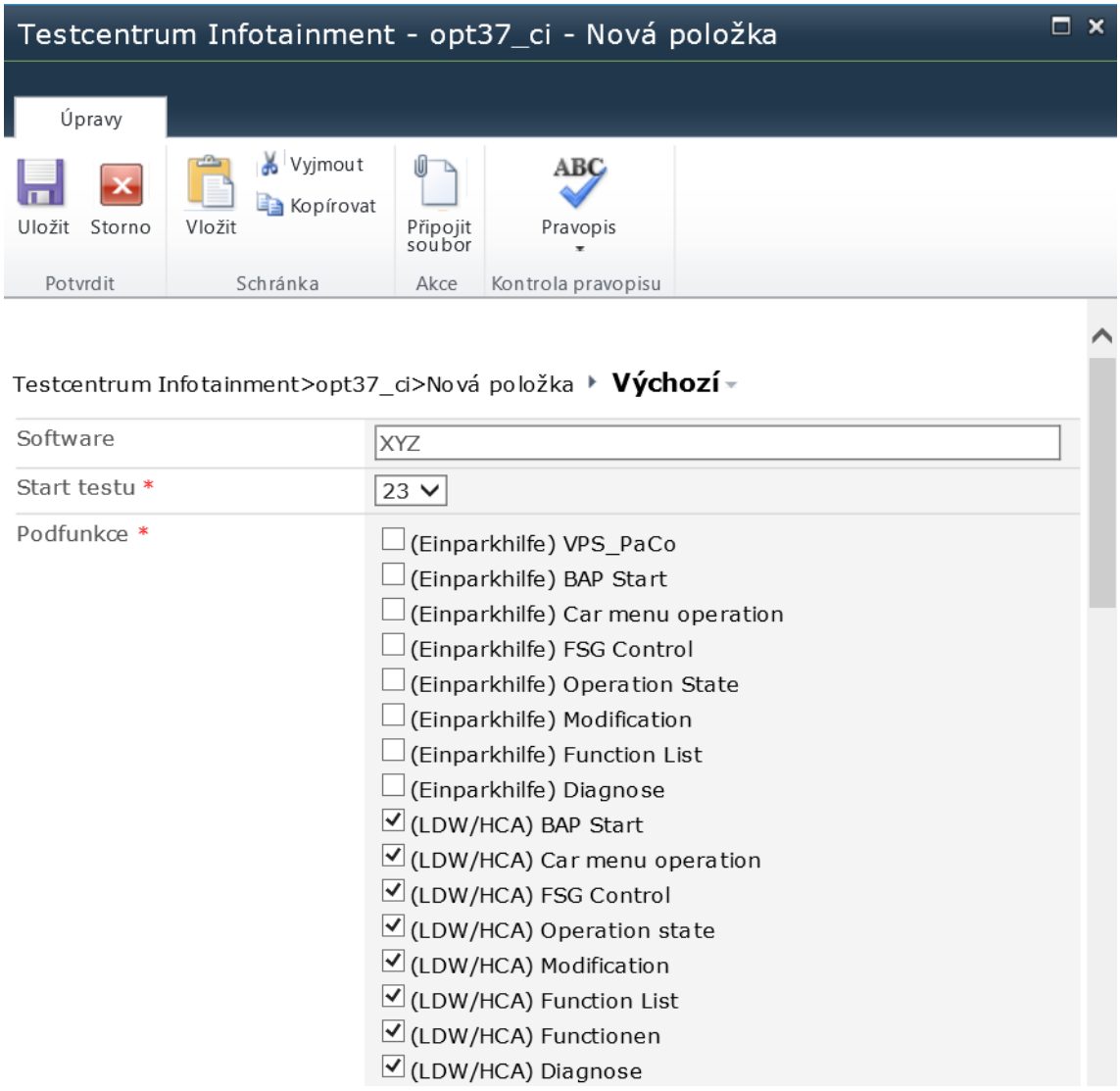
Po kliknutí na odkaz „Zadat test“ v hlavním menu webu je návštěvník přesměrován na stránku s nabídkou řídicích jednotek infotainmentu. Aktuálně se jedná o sedm řídicích jednotek, z nichž každá má kolem patnácti testovatelných funkcí. Kliknutím na řídicí jednotku se zobrazí její funkce. Požadavek na test se vždy zadává pro jednu funkci jednotky (podání požadavku má na starosti tzv. funkční vlastník). Po kliknutí na zvolenou funkci se načte stránka s komponentou Teamwebu, tzv. seznamem. Seznam je komponenta umožňující shromažďovat data zadaná uživatelem v tabulce s vlastními definovanými sloupci. Data se zadávají prostřednictvím formuláře kliknutím na tlačítko „nová položka“.

Množství seznamů je prvním problémem Teamwebu. Pokud přijde do testování nová řídicí jednotka, je třeba vytvořit počet HTML stránek odpovídající počtu funkcí řídicí jednotky. Pro každou stránku je pak třeba vytvořit vlastní seznam k uchovávání požadavku. Konfigurace jedné komponenty seznamu a udělení oprávnění uživatelům na přístup k seznamu je časově poměrně náročnou záležitostí (10-15 minut). Řekněme, že má nová řídicí jednotka patnáct funkcí, potom vytvoření obsahu pro zadání požadavku pro tuto jednotku zabere orientačně 15 × 15 minut, tedy necelé čtyři hodiny rutinní, repetitivní práce. V případě, že je potřeba provést více změn u všech sedmi stávajících řídicích jednotek (například při změně testovacích specifikací), může úprava webu zabrat tři až čtyři pracovní dny.

Problém s množstvím seznamů by byl ve vlastní implementaci webové aplikace velmi dobře řešitelný. Požadavky by byly zadávány pomocí jediného formuláře, ze kterého by byla data ukládána do vhodné databázové entity. Řídicí jednotku i funkci by zadavatel mohl zvolit přímo ve formuláři. Při nutnosti přidat novou řídicí jednotku do testování by poté stačilo přidat nové záznamy s názvem jednotky a jejích funkcí do databáze. Tato změna by výrazně snížila dobu potřebnou pro údržbu aplikace. Změna, která ve stávající aplikaci trvá řádově několik hodin, by v novém řešení zabrala několik minut.

Dalším problémem, který není možné vyřešit prostřednictvím aplikace Teamweb, je nutnost zobrazovat veškerá podtémata k zaškrtnutí pro test při vyplňování formuláře. Formulář není možné sestavit dynamicky tak, aby se zobrazovala pouze podtémata funkce

na základě toho, zda bylo zaškrtnuto odpovídající funkční téma. Všechna funkční podtémata jsou tedy vypsána naráz, při čemž názvy témat jsou uvedeny v závorce (viz Obrázek 2), vzniká tak dlouhý a nepřehledný formulář, kterým je nutné scrollovat.



Obrázek 2: Část aplikace pro zadání požadavku na test v aplikaci Teamweb

Ve vlastním řešení je možné tento nedostatek odstranit vytvořením dynamického formuláře, který zobrazí pouze hlavní funkční témata funkce (uvedená v závorkách na Obrázek 2). Při zaškrtnutí tématu by se odkryla všechna jeho funkční podtémata. Formulář v této podobě by byl mnohem přehlednější a jeho vyplňování rychlejší.

Poslední nežádoucí vlastností Teamwebu je, že sama aplikace nemůže vygenerovat testovací specifikace pro testování na základě zadaného požadavku. Požadavek musí být

nejprve vyexportován do souboru formátu XLS, ze kterého se následně pomocí nástroje Test Planner vytvoří testovací specifikace.

2.2 dTCM

Desktopový nástroj dTCM, celým názvem Doors Test Case Manager, je určený k manuálnímu testování. Jeho primární funkcí je zobrazovat v textové podobě testovací případy. Tester podle instrukcí provede zkoušku a následně testovací případ pomocí tlačítek ohodnotí podle toho, zda byl výsledek zkoušky úspěšný nebo neúspěšný. V případě, že výsledek přesně neodpovídá očekávanému výsledku popsanému v testovacím případu, musí tester vyplnit chybový formulář a následně chybu zadat do aplikace KPM (nástroj pro evidenci nalezených chyb).

2.2.3 Nevýhody dTCM

Aplikace dTCM používá jako vstup pro testování testovací specifikaci soubor ve formátu DXML. Tester si před testováním nahrává požadované DXML soubory do odpovídajícího adresáře v předem připravené struktuře (adresáře jsou rozděleny podle projektů a kalendářních týdnů).

V aplikaci dTCM tester vyhodnocuje jednotlivé testovací případy a výsledek je zaznamenán opět do stejného DXML souboru. Nevýhodou takového ukládání výsledků je, že každý, kdo má přístup do daného adresáře na disku, může ať už úmyslně, nebo omylem výsledky změnit. Další nevýhodou je narůstající adresářová struktura s historickými výsledky, která zabírá stále více místa na disku. Zároveň se snižuje přehlednost uspořádání dat a vyhledávání konkrétních záznamů z testů je s přibývajícím časem náročnější.

Problém s neoprávněným zásahem do výsledků testů je ve vlastní aplikaci možné řešit ukládáním výsledků ukončených testů do relační databáze. Ta v tomto případě přináší několik výhod. Data jsou lépe zabezpečena a další manipulace s nimi je výrazně jednodušší a rychlejší oproti práci s mnoha textovými soubory.

2.3 TErU

TErU – Test Ergebnisse Übersicht (v překladu přehled výsledků testu) je, jak název napovídá, aplikace pro zobrazování výsledků testů. Zdrojem dat pro aplikaci TErU jsou DXML soubory s testovací specifikací dříve otestovanou v aplikaci dTCM.

V TErU je možné filtrovat výsledky testů podle projektu, kalendářního týdne a testovacího místa. Aplikace nabízí několik režimů zobrazení. Základní typy režimů jsou *chart* a *view*.

Zobrazení grafů je potom možné přepínat mezi režimy *Function chart*, *Group chart* a *Sum chart* (přehled výsledků podle funkcí, seskupení sdružených funkcí a součet všech provedených testovacích případů). Jedná se o sloupcové grafy, ukazující příslušnou barvou počet testovacích případů, které prošly testem, neprošly testem a nebyly otestovány.

Function view, Week view, Unit view, All Units view, Project view a KPM view jsou specifické přehledy výsledků testů, podle daného kritéria.

2.3.4 Nevýhody TErU

TErU velmi dobře předkládá uživateli výsledky testů v grafické podobě. Nabízí všechny potřebné funkce k získání přehledu o aktuálním i dříve uskutečněném testování. K dispozici je množství různých filtrů dat, rozmanité typy pohledů na výsledky i nalezené chyby a chybové tickety. Slabým místem TErU je opět formát dat, se kterým aplikace pracuje. Množství souborů, jejich umístění a velikost mají negativní vliv na plynulý chod TErU. Přehledy se zobrazují na základě dat získaných z uložených DXML souborů. Tato data jsou nejdříve načtena do vlastní SQLite databáze, takže prohlížení historických dat je bezproblémové, ale při aktualizaci a načítání nových dat dochází často k zamrznutí nebo i pádu aplikace.

Ve vlastním řešení by se mohly výsledky testů ukládat do relační databáze přímo během testování. Při prohlížení výsledků by potom nebylo potřeba průběžné načítání velkého množství dat z DXML souborů, které zpomaluje stávající aplikaci.

2.4 KPM

Aplikace KPM – Konzern Problem Management umožňuje uživatelům identifikovat, zaznamenávat a upravovat chyby (problémy) napříč koncernovými značkami a obchodními jednotkami podle definovaných předpisů. Zároveň je možné sledovat stav řešení problému včetně analýz a podniknutých opatření. V případě, že testovací případ neprojde testem, zadává se nalezená chyba právě do systému KPM.

2.5 IBM Rational Doors

Aplikaci IBM Rational Doors využívá s vlastní nástavbou Testcentrum elektroniky pro správu a export testovacích případů. Testovací případy jsou z Doors exportovány ve formátu DXML.

Rational Doors je nástroj pro správu požadavků, který usnadňuje zachycení, sledování, analýzu a správu změn informací. Doors je zkratka pro Dynamic Object-Oriented Requierements System [4].

K databázi požadavků je možné přistupovat prostřednictvím webového rozhraní. Tyto požadavky je možné propojovat s plány testů a testovacími případy. Uživatelé mohou komunikovat změny v diskuzi.

2.6 Vyhodnocení

Test management nástroje v Testcentru elektroniky jsou vzájemně provázané. Podle předchozí rešerše je možné stávající nástroje rozdělit na dvě skupiny. Do první z nich patří nástroje, které jsou určeny přímo a pouze pro potřeby testování. Druhá skupina zahrnuje nástroje používané i v jiných odděleních napříč koncernem Volkswagen Group. Tato oddělení používají nástroje i pro jiné účely než testování.

Do první skupiny z dříve zmíněných nástrojů patří Teamweb Testcentrum Infotainment, dTCM a TErU. Všechny tři nástroje jsou závislé na datech, která jsou uložena v Doors. Jejich používání a výstupy však nemají přímý vliv na další nástroje používané v rámci koncernu a jejich funkce slouží spíše ke zjednodušení procesu testování na lokálním pracovišti ve Škoda Auto. Z tohoto důvodu je možné v této skupině provádět změny nástrojů, v případě, že některé funkce jsou nevyhovující.

Rešerše ukázala, že nejvýznamnějším problémem současného softwarového vybavení je absence společné datové struktury, kterou by mohli snadno využívat všechny nástroje. Důsledkem je nutnost manipulovat ručně s daty a v mezikrocích používat utility pro jejich úpravu před předáním další aplikaci. Jako řešení tohoto problému jsem navrhl vytvoření nové webové test management aplikace, která zastoupí důležité funkce stávajících nástrojů a bude používat jednotnou relační databázi. Funkce, které nejsou z uživatelského hlediska v současnosti optimální, nově vzniklá aplikace zdokonaluje. Podrobný popis nové aplikace včetně návrhu se nachází v kapitole 4 a dále.

Druhá skupina nástrojů, do které z dříve zmíněných patří KPM a Doors, není vhodná pro žádnou úpravu. Tyto nástroje jsou pevnou součástí procesu v koncernu a jejich náhrada by byla velmi náročná. Funkce obou aplikací se navíc dlouhodobě osvědčila, aplikace jsou spolehlivé, robustní a Doors také podléhá stálé údržbě IBM.

3 Existující test management software na trhu

Před vývojem vlastní aplikace jsem provedl rešerši existujících řešení. Aplikace zaměřené na test management obvykle cílí na testování softwaru a často nenabízí podporu pro řízení komplexnějšího testování, jakým zkoušky elektroniky vozidla jsou.

V potaz musí být bráno několik proměnných navíc, například verze hardwaru řídící jednotky nebo druh testu podle testovacího místa. Řídicí jednotky ve voze spolu komunikují, a tak musí být při testování přítomen další hardware (nebo jeho simulace), na kterém je testovaná jednotka závislá. Podpora pro spravování těchto konfigurací je v dostupných test management systémech výjimečná.

Nástroje pro plánování testu často nevyhovují specifickému testovacímu procesu oddělení Testcentra elektroniky. V následující podkapitole jsou popsána kritéria, na základě kterých jsem vybral nástroje pro podrobnější popis.

3.1 Kritéria rešerše

Test management aplikací existuje na trhu velké množství, často však neodpovídají potřebám pro testování fyzických řídicích jednotek. Volbu nástrojů, kterým se dále v práci věnuji, jsem provedl na základě toho, zda jsou jejich funkce relevantní vzhledem k testovacímu procesu v Testcentru elektroniky Škoda Auto.

Nejdříve jsem vytvořil širší výběr aplikací na základě porovnání tří různých aktuálních žebříčků nejlepších test management nástrojů (k roku 2019). Do širšího výběru jsem zvolil aplikace, které se objevily alespoň ve dvou žebříčcích a zároveň se umístily co nejvýše. První použitý žebříček pochází z webu Software Testing Help [5]. Druhým je žebříček z Guru99 [6]. Posledním zdrojem je žebříček ze Software Testing Material [7].

Seznam vybraných test management aplikací:

- IBM Rational Quality Manager
- Microfocus Application Lifecycle Manager
- Zephyr
- TestRail

- qTest
- PractiTest
- Test Collab
- TestCaseLab
- XQual
- Adaptavist

Do užšího výběru aplikací, které dále popisuji, jsem vybral ty, které přinášejí nějakou významnou výhodu při použití v prostředí Škoda Auto. Jedná se například o možnost snadné integrace nového nástroje s některou již používanou aplikací ve firmě, podporu pro konfiguraci prostředí a podmínek testu nebo široké možnosti vlastní customizace test management nástroje.

3.2 IBM Rational Quality Manager

IBM Rational Quality Manager (RQM) je webová aplikace pro komplexní plánování a vytváření testů a také testování funkcí v průběhu celého životního cyklu vývoje. Tento nástroj je určený primárně k testování softwaru. RQM podporuje různé role uživatelů jako je například test manažer, test architekt, tester a další. Podporovány jsou také role mimo testovací organizaci.

Ve fázi plánování RQM nabízí širokou škálu možností. Test manager definuje cíle projektu, zavádí kontrolní schvalovací proces pro plán zkoušek a testovací scénáře. Stanovuje závislosti mezi testovacími scénáři a požadavky na projekt, které musí být pro provedení testu splněny. Součástí plánování je i odhad náročnosti konkrétního testu. Časový plán je možné definovat pro každou iteraci testu. Uživatel zadává možná prostředí pro test, z nichž jsou nástrojem generovány testové konfigurace. Dále se určují například nároky na kvalitu, vstupní podmínky a kritéria pro ukončení testu. RQM nabízí také možnost vytváření a správy testovacích případů a sledování průběhu prováděných testů.

K návrhu testovacích případů slouží *rich text* editor. Prostřednictvím editoru je možné přidat odkazy na veškeré informace o pozadí testu (nutné požadavky, předmět testování,

konfigurace). Testovací případ lze asociovat s konkrétním test plánem a záznamem o provedení testu. Testovací případy mohou být kombinovány do rozsáhlejších souborů zkoušek.

RQM podporuje správu a spouštění testovacích skriptů. Tyto skripty ovšem musí být vytvořeny pomocí nástrojů od IBM (Rational Functional Tester, Rational Performance Tester, Rational Robot a další). Testcentrum elektroniky používá v současnosti vlastní automatické testy, které není možné vzhledem k omezení na nástroje od IBM do systému RQM integrovat.

V prostředí pro běh testů je k dispozici několik nastavení. Testován může být samostatný testovací případ nebo skupina testovacích případů. Tyto skupiny testů mohou běžet sekvenčně nebo současně v paralelním módu.

RQM obsahuje set předdefinovaných reportů, které napomáhají sledovat současný stav projektu. Je možné doinstalovat volitelnou komponentu Rational Reporting for Development Intelligence nebo Rational Insight, zmíněné komponenty nabízí další typy reportů a možnost customizace reportů pro konkrétní potřeby uživatele.

Mezi výhody patří:

- jednoduchá integrace s již používaným systémem Rational DOORS,
- podpora pro konfiguraci testovacího prostředí.

Nevýhodami jsou:

- nedostatečně obsáhlé možnosti pro plánování projektu,
- nedostatečná přizpůsobitelnost systému pro specifické potřeby Testcentra,
- zpětná vazba uživatelů na zákaznické podpoře je často negativní (časté chybové stavy, pomalá odezva).

3.3 Microfocus Application lifecycle management

ALM byl původně produkt společnosti Mercury, který později převzala společnost Hewlett-Packard. Nyní vlastní ALM firma Microfocus. Nástroj prošel při změně majitele patrným vývojem, ale od svého počátku je stejně jako RQM stále webovým nástrojem. Opět se jedná o produkt, který je především zaměřený na testování softwaru.

Nabídka funkcí ALMu je podobná té, kterou je možné nalézt v RQM. Zřejmou výhodou je modernější a přívětivější uživatelské rozhraní. Pro komunikaci v týmu využívá ALM aplikaci Skype for Business. Skype je ve Škoda Auto (vedle klasického e-mailu) standardním prostředkem komunikace mezi zaměstnanci, z tohoto hlediska je jeho využití a podpora v ALMu nespornou výhodou. Základní software je možné rozšířit o další komponenty, například modul Load Runner, který umožňuje testovat výkon aplikace prostřednictvím virtuálních uživatelů, kteří zatěžují server požadavky. Tyto moduly jsou opět ve většině případů vhodné pro testování softwaru a pro Testcentrum elektroniky nepřinášejí užitnou hodnotu.

Doporučeným prohlížečem je Microsoft Internet Explorer 10 32 Bit (a verze 11 32 Bit), který je také výchozím prohlížečem na všech firemních počítačích ve Škoda Auto. Aplikace má také své mobilní verze pro všechny přední platformy (Windows 7, 8.1, 10; Android 5.0, 5.1, 6.0, 7.0; iOS 10.3.x, 11)

Mezi výhody patří:

- podpora pro konfiguraci testovacího prostředí,
- přehlednější uživatelské rozhraní,
- verze pro mobilní zařízení,
- využívá Skype for Business.

Nevýhodami jsou:

- nedostatečně obsáhlé možnosti pro plánování projektu,
- nedostatečná přizpůsobitelnost systému pro specifické potřeby Testcentra,
- opět velké množství negativní zpětné vazby od uživatelů. Ta se týká zejména pomalé odezvy systému, častého zamrzání a nedokonalé dokumentace.

3.4 Zephyr

Jedná se o další webový nástroj, který by měl přispět ke zdokonalení procesu testování. Stejně jako u dříve zmíněných nástrojů platí nevýhoda v jeho orientaci na vývoj a testování softwaru, které se od testování řídících jednotek automobilu v několika ohledech liší.

Funkčnost Zephyru je z části závislá na aplikaci JIRA, kterou používají týmy vyvíjející software k udržování přehledu o stavu projektu (verze, přidělování úkolů, chyby atd.). Zephyr se distribuuje ve dvou podobách.

Zephyr for JIRA (Native Jira Edition) se instaluje pouze jako plug-in přímo do aplikace JIRA. Díky tomuto provázání má testovací plug-in okamžitý přístup k datům z vývoje. Zephyr for JIRA je určen především pro menší týmy, které začínají s test managementem a chtějí veškeré testování spravovat skrze aplikaci JIRA.

Oproti tomu Zephyr (Standalone Edition) je plnohodnotným test management softwarem s podporou pro automatické testování a rozšířenou funkcionalitou. Tato verze je funkčně srovnatelná s ostatními porovnávanými nástroji, navíc nabízí funkce pro správu a report chyb. Zephyr má širokou podporu integrace dalších osvědčených testovacích nástrojů (Jenkins, Selenium, UFT), ty jsou ovšem opět určeny pouze pro testování softwaru.

Mezi výhody patří:

- integrace s aplikací JIRA,
- správa chyb,
- intuitivní, snadné použití.

Nevýhodami jsou:

- chybějící podpora pro konfiguraci prostředí testu,
- pevně stanovený formát podoby testovacích případů.

3.5 TestRail

TestRail je další aplikací v řadě, která umožňuje integraci s JIROU, ale také mnoha dalšími nástroji. Stejně jako v předchozích případech se jedná o webovou aplikaci. Silnou stránkou tohoto softwaru je detailní správa testovacích případů a uživatelsky přehledná organizace testů do sekcí a složek. Na druhou stranu testovací případy není možné snadno sdílet mezi projekty.

Zdařilou je také část aplikace orientovaná na reporting. Kromě standardních grafů, které ukazují výsledky testů, nabízí TestRail sledování vytížení jednotlivých členů týmu (testerů), což umožňuje efektivnější rozdělování práce.

Největší výhodou oproti konkurenci je vysoká úroveň customizace celé aplikace. TestRail dovoluje uživateli upravovat podobu testovacích případů (přejmenovávat a přidávat vlastní pole). Tyto změny se mohou projevovat pouze v konkrétním projektu nebo i globálně. Je možné přidávat vlastní role uživatelů na míru daného testovacího týmu a následně přidělovat oprávnění.

Aplikace má editor pro skriptování, skrze který se dá upravovat uživatelského rozhraní a některé další části aplikace. Díky REST API, možnosti implementovat vlastní systém autentizace a vysoké míře customizace, je TestRail vhodný i pro nestandardního zákazníka, který není orientován pouze na testování softwaru.

Mezi výhody patří:

- vysoká míra customizace,
- mnoho možností v oblasti plánování,
- reporting,
- jednoduché použití.

Nevýhodami jsou:

- absence přímé podpory konfigurace prostředí testu,
- nemožnost oddělit testovací případy pro manuální a automatické testy.

3.6 Vyhodnocení

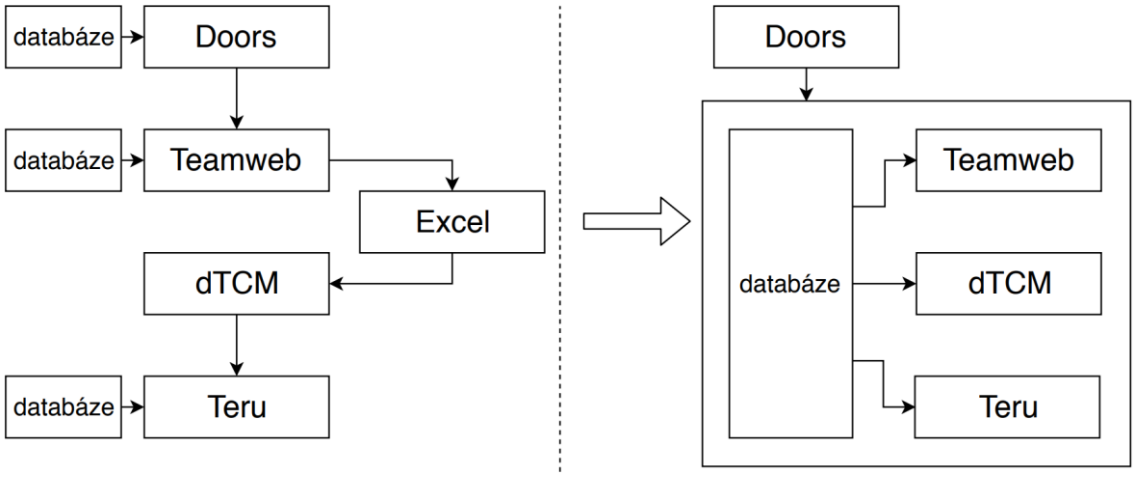
Existující test management nástroje nejsou dostatečně obecné pro snadné použití v Testcentru elektroniky. Současná řešení cílí převážně na týmy vyvíjející software pro standardní osobní počítače a další masově používané platformy, postrádají ale možnost detailnějšího popisu závislostí, které musí být splněny pro otestování některých řídicích jednotek. Obecnost nástrojů nemusí být problémem pro menší týmy vývojářů, které jsou do jisté míry schopné přizpůsobit proces testování používanému nástroji, v případě rozsáhlého specializovaného pracoviště toto možné není.

4 Návrh řešení

Jako řešení současných nedostatků v oblasti test management nástrojů Testcentra elektroniky jsem navrhl tvorbu vlastní webové aplikace, která bude odpovídat testovacímu procesu na tomto pracovišti. Jak vyplývá z rešerše, vývojáři většiny test management nástrojů využívají právě formu webové aplikace. Ta je výhodná z několika důvodů. Jedním z nich je, že koncový uživatel nemusí aplikaci instalovat. K používání stačí webový prohlížeč, který je již nainstalován v operačním systému každého firemního počítače. Aplikace je stále *online* a všichni uživatelé mají vždy aktuální verzi. Další výhodou je bezpečnost a jednoduchá správa centralizované databáze.

Myšlenkou nového řešení je pokrytí funkcí některých stávajících nástrojů v rámci jedné ucelené aplikace a zároveň používání jediného zdroje dat. Vlastní řešení navíc umožňuje upravit současné funkce, které nejsou vyhovující.

Na obrázku níže (viz Obrázek 3) je vidět schematické porovnání současného a nového přístupu. V současnosti je třeba nejdříve získat názvy nových funkcí, funkčních témat a podtémat pro aplikaci Teamweb z testovacích specifikací v aplikaci Doors. Data jsou ve formátu DXML a vkládají se ručně do databáze Teamwebu. Po zadání testu na Teamwebu musí být požadavek exportován do aplikace Excel. Výslednou tabulku poté zpracuje nástroj pro úpravu testovacích specifikací, který vytvoří novou specifikaci obsahující pouze testovací případy zadané v požadavku.



Obrázek 3: Srovnání současného přístupu (vlevo) a nového přístupu (vpravo)

Vytvořená testovací specifikace ve formátu DXML může být dále načtena aplikací pro provedení testu (dTCM). Při testování se výsledky ukládají do téhož DXML souboru, který následně používá na vstupu aplikace TErU pro zobrazování výsledků (podrobnější informace o jednotlivých nástrojích obsahuje kapitola 2).

Nové řešení počítá s jediným externím zdrojem dat, a to jsou testovací specifikace z aplikace Doors, kterou není momentálně možné z interních důvodů firmy v procesu nahradit. Specifikace se mění a vznikají nové, je tedy nutné v určitých časových intervalech databázi nové aplikace aktualizovat.

V pravé části obrázku na předchozí straně (viz Obrázek 3) je možné vidět zjednodušené schéma nové aplikace. Její funkční části jsou ve schématu pro názornost pojmenovány názvy odpovídajících nástrojů. Nová aplikace používá pro zadávání požadavku, testování i zobrazování výsledků jedinou databázi.

Díky jedné databázi je ušetřena režie spojená s nutností přesunu dat z jedné aplikace do druhé. Další výhodou související s tímto přístupem je rychlejší odezva, protože veškerá aktuální data jsou již přímo v databázi a není třeba je před výpisem nejprve načítat ze souborů na disku.

4.1 Zvolené funkce

Konkrétní zvolené funkce nové aplikace, které vyplývají z rešerše, byly konzultovány s koordinátorem testování infotainmentu a dalšími zaměstnanci na oddělení. V nejobecnější rovině návrhu jsou požadovány čtyři hlavní funkce:

- zadávání požadavků na test,
- správa uložených testovacích případů,
- manuální testování,
- prezentace výsledků v podobě grafů.

Kromě čtyř uvedených hlavních funkcí je požadováno, aby aplikace využívala systém uživatelských účtů. Uživatelé mají role a s nimi spojená oprávnění. Další požadovanou funkcí je administrační část aplikace, kde uživatel s oprávněním administrátora může jednoduše spravovat databázi. Podrobnější požadavky na hlavní funkce jsou pospány v podkapitolách dále.

4.1.1 Zadávaní požadavků na test

V části aplikace pro zadávání požadavků na test je možnost vytvářet, editovat a prohlížet požadavky. Vytváření požadavků se skládá z následujících informací: název řídicí jednotky, požadovaná funkce, funkční témata a podtémata, verze softwaru řídicí jednotky (případně i verze hardwaru), číslo kalendářního týdnu, ve kterém má proběhnout test, poznámka funkčního vlastníka, poznámka test manažera a informace o tom, zda byl test zapracován do test plánu, či nikoliv.

Na stránce se seznamem všech zadaných požadavků je možnost jejich filtrování podle atributů uvedených v odstavci výše. Seznam je také možné exportovat do aplikace Excel, pro kterou již existují nástroje, které umí z dat automaticky vytvořit grafické podklady pro plánování.

4.1.2 Správa testovacích případů

Ve správě testovacích případů je možnost nahrát DXML testovací specifikaci z disku počítače do databáze aplikace. Testovací případy je možné vytvářet, editovat i mazat a mají stejnou formu jako ve stávajících testovacích specifikacích. Testovací případy lze filtrovat podobně jako požadavky na test, podle názvu, související řídicí jednotky atd.

4.1.3 Manuální testování

Pro manuální test je k dispozici seznam požadavků. Tester může vyhledávat příslušný požadavek podle funkce a verze softwaru, kterou má otestovat. Po zahájení testu se načtou odpovídající testovací případy, které byly zvoleny zadavatelem testu. Každý testovací případ je možné vyhodnotit jako splněný, nesplněný, nebo označit varováním. Při nalezení chyby je možné ji zařadit do předdefinovaných kategorií, popsat ji a komentář uložit. Testovací případ může být přeskočen a označen jako netestovaný. Tester má k dispozici měření času pro případ časově závislých testovacích případů.

4.1.4 Prezentace výsledků

Výsledky jsou prezentovány v podobě sloupcových grafů, které nesou informaci o tom, kolik testovacích případů testem prošlo, kolik neprošlo a kolik jich bylo netestovaných. Výsledky je možné prohlížet na úrovní funkcí a řídicích jednotek. Pro zobrazení výsledků konkrétního testu je k dispozici vyhledávání.

4.2 Zvolené technologie

Pro serverovou část aplikace jsem zvolil framework Django, pro který jsem se rozhodl z několika důvodů. Prvním z nich je automatické generování administračního rozhraní. Django používá metadata z modelové vrstvy aplikace, ze kterých vytváří rozhraní správy databáze pro uživatele s příslušným oprávněním [8]. Jedním z požadavků na novou aplikaci je vytvoření právě administrační části. Generovaná Django administrace umožňuje více se soustředit na vývoj vlastní aplikace a šetří rutinní práci s vytvářením backendu.

Další výhodou vybraného frameworku je bezpečnost. Django poskytuje nástroje k vytváření bezpečných webových aplikací a samotný framework má implementované některé bezpečnostní prvky, jako je například prevence spouštění cizího kódu na úrovni šablon.

Django patří mezi nejpoužívanější webové frameworky. V průzkumu Stack Overflow se umístil na osmém místě mezi ostatními webovými frameworky (zahrnuty byly frontendové i backendové frameworky) [9]. Díky jeho popularitě a početné komunitě je snazší řešení problémů v průběhu vývoje.

Pro uchovávání dat jsem zvolil relační databázový systém PostgreSQL. Systém je odpovídající vzhledem k rozsahu aplikace a množství strukturovaných dat, se kterými pracuje. Rozhodujícím důvodem pro volbu PostgreSQL byla také jeho plná podpora ve frameworku Django. Podle průzkumu Stack Overflow je PostgreSQL druhý nejpoužívanější databázový systém současnosti [9].

Jako frontendovou technologii jsem zvolil jazyk JavaScript a knihovnu JQuery. Povaha aplikace nevyžaduje použití robustnějšího frontendového frameworku. Pro jednotný design aplikace jsem zvolil framework Bootstrap.

4.2.5 Django

Django je open-source webový framework napsaný v jazyce Python. Používá návrhový vzor model-view-template (MVT) [10], který se podobá známé architektuře model-view-controller (MVC). Rozdílem je, že framework Django sám obstarává část, která se nazývá controller. V případě MVT model představuje strukturu a logiku práce s daty, view popisuje, jaká data budou prezentována, a template vrstva určuje, jak budou data prezentována.

Framework udržuje nezávislá nezisková organizace Django Software Foundation. Aplikace používá Django ve verzi 2.1.

4.2.6 PostgreSQL

PostgreSQL je open-source databázový systém typu RDBMS (relational database management system). Je to výchozí databáze pro macOS Server [11], zároveň je dostupná pro Linux i Windows. Aplikace používá PostgreSQL ve verzi 10.5.

4.2.7 jQuery

JavaScript knihovna jQuery slouží především ke snazší manipulaci s objektovým modelem dokumentu (DOM). DOM je stromová struktura reprezentující veškeré elementy webové stránky [12]. Příkladem využití této knihovny je hledání elementu s určitou vlastností, změna atributů vybraného elementu nebo nastavení reakce elementu na událost (např. kliknutí).

4.2.8 AJAX

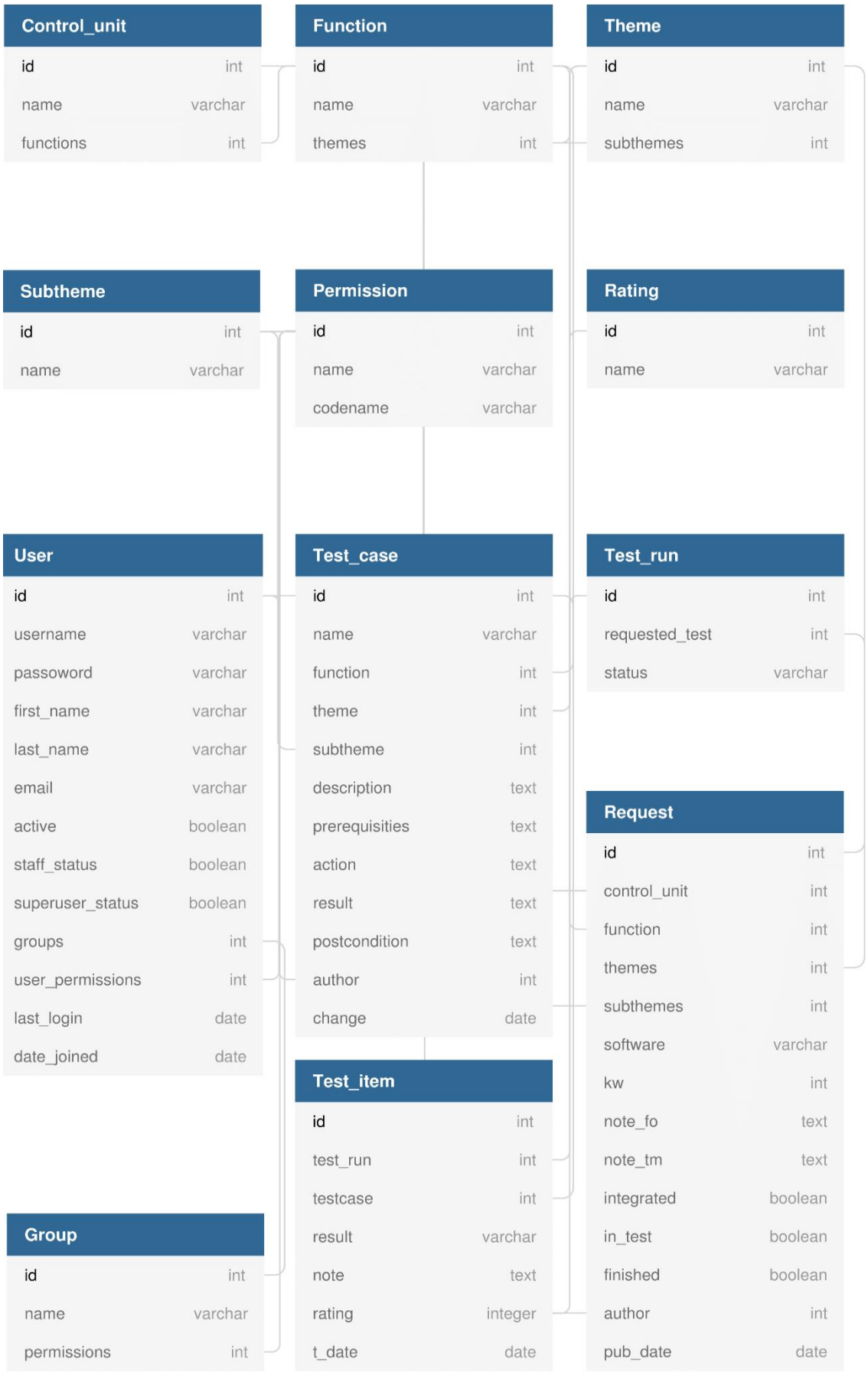
Další z použitých technologií je AJAX (zkratka pro Asynchronous JavaScript and XML). AJAX umožňuje odesílání a opětovné získávání dat ze serveru bez nutnosti znovunačtení stránky.

4.2.9 Bootstrap

Bootstrap je open-source framework obsahující předdefinované komponenty uživatelského rozhraní založené na CSS a jazyce JavaScript. Bootstrap komponenty jsou responzivní, díky tomu je možné aplikaci jednoduše používat na různých velikostech displeje.

4.3 Návrh databáze

Při návrhu databáze jsem vycházel z funkčních požadavků. Názvy entit jsou pro přehlednost zvýrazněny tučně a názvy atributů kurzívou. Pro funkci zadávání požadavků na test je důležitá entita popisující požadavek – **Request** (přehled všech navržených entit a jejich atributů včetně datových typů viz Obrázek 4 na další straně). Jejím primárním klíčem je *id*, jedná se o umělý primární klíč, který jsem zvolil i pro všechny ostatní entity.



Obrázek 4: Návrh databáze

Entita **Request** je popsána atributy, které nesou informace o tom, co chce zadavatel požadavku testovat. Atributy pro řídicí jednotku, funkci, funkční téma, funkční podtéma a autora požadavku jsou cizími klíči, které tvoří následující vazby s uvedenými entitami (vazba je popsána v pořadí cizí klíč, odkazovaná entita, kardinalita vazby):

- *control_unit*, **Control_unit** (N:1),
- *function*, **Function** (N:1),
- *themes*, **Theme** (M:N),
- *subthemes*, **Subtheme** (M:N),
- *author*, **User** (N:1).

Test_case je entita, která reprezentuje testovací případ. Ten obsahuje atributy, které popisují jeho zařazení v testovací specifikaci (funkce, funkční téma, funkční podtéma), dále atributy definující samotný test (název testovacího případu, popis testu, počáteční podmínky, akce, očekávaný výsledek, podmínky po skončení testu) a nakonec atributy o čase poslední změny a jejím autorovi. **Test_case** tvoří vazby s následujícími entitami:

- *function*, **Function** (N:1),
- *theme*, **Theme** (M:N),
- *subtheme*, **Subtheme** (M:N),
- *author*, **User** (N:1).

Entita **Test_runs** je záznam o provedeném nebo prováděném testu. Jedním z jejích atributů je cizí klíč *requested_test*, který odkazuje na požadavek, na jehož základě se testuje. Dalšími atributy jsou *status* a *t_date*. Atribut *status* udává, zda test běží, nebo byl již ukončen a *t_date* je čas ukončení testu. Entita **Test_run** tvoří tuto vazbu:

- *requested_test*, **Request** (1:1).

Entita **Test_item** popisuje vlastnosti jedné položky testu. Jedná se o záznam výsledku testu konkrétního testovacího případu v dané instanci. **Test_item** obsahuje cizí klíče, které odkazují na příslušný běh testu a testovací případ. Kromě výsledku testu nese také informaci o případné chybě a její závažnosti (atributy *note* a *rating*). **Test_item** tvoří vazbu s entitami **Test_run**, **Test_case** a **Rating**:

- *test_run*, **Test_run** (N:1),
- *testcase*, **Test_case** (N:1),
- *rating*, **Rating** (N:1).

Entita **Users** představuje uživatele aplikace. Pro přihlášení do systému slouží atributy *username* a *password*, dále entita uchovává osobní údaje uživatele (*first_name*, *last_name*, *email*) a informace spojené s jeho účtem, jako jsou čas posledního přihlášení nebo status. Uživatelé jsou řazeni do skupin a mají přidělená oprávnění:

- *groups*, **Group** (M:N),
- *user_permissions*, **Permission** (M:N).

Group je entita popisující skupiny uživatelů, tyto skupiny mohou mít přidělena specifická oprávnění stejně jako jednotliví uživatelé. Entita **Permission** slouží k popisu oprávnění v rámci aplikace.

Entita **Control_unit** zastupuje řídicí jednotku, která může mít několik funkcí (*functions)*. **Function** může obsahovat více funkčních témat (*themes*) a entita **Theme** se může skládat z několika funkčních podtémat (*subthemes*). Entita **Subtheme** obsahuje pouze svůj vlastní název (*name*).

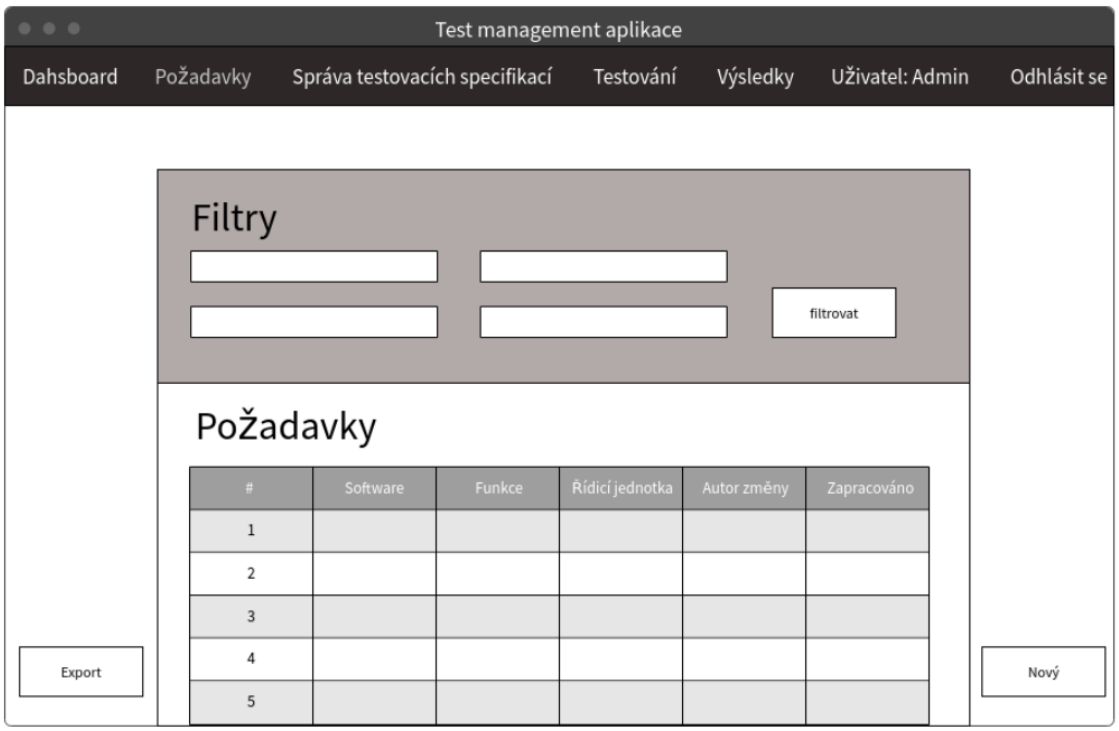
4.4 Uživatelské rozhraní

Uživatelské rozhraní jsem navrhoval pomocí webového nástroje MockFlow. Ten umožňuje rychle vytvářet základní rozvržení stránek a jejich prvků přetahováním předdefinovaných elementů na pracovní plochu. Vzniklé návrhy je díky tomu snadné okamžitě upravovat a přizpůsobovat potřebám uživatele.

Při návrhu jsem kladl důraz na jednoduchost a snadnou orientaci v aplikaci. Volil jsem takové rozložení elementů, na které jsou uživatelé zvyklí při používání většiny běžných webových aplikací.

Pro navigaci po webové aplikaci jsem zvolil v současnosti nejobvyklejší horizontální menu v horní části okna prohlížeče. Položky levé části menu tvoří hlavní funkční celky aplikace. V pravé části menu je funkce pro přihlášení/odhlášení uživatele, případně jméno

přihlášeného uživatelského účtu. Veškerý prostor pod menu je vyčleněný pro vlastní obsah aktuální stránky.



Obrázek 5: Návrh rozložení stránky s požadavky na test

Aplikace je určena pro používání na osobních počítačích, které mají k dispozici všichni zaměstnanci, odpovídají tomu i všechny navržené šablony. Díky použití knihovny Bootstrap je aplikace sice responzivní, ale jako celek nebyla pro používání na mobilních zařízení navržena. Například zadání požadavku je možné pohodlně provést i z mobilního telefonu, oproti tomu jiné funkce (testování nebo prohlížení výsledků) jsou ze své podstaty nevhodné pro malé displeje.

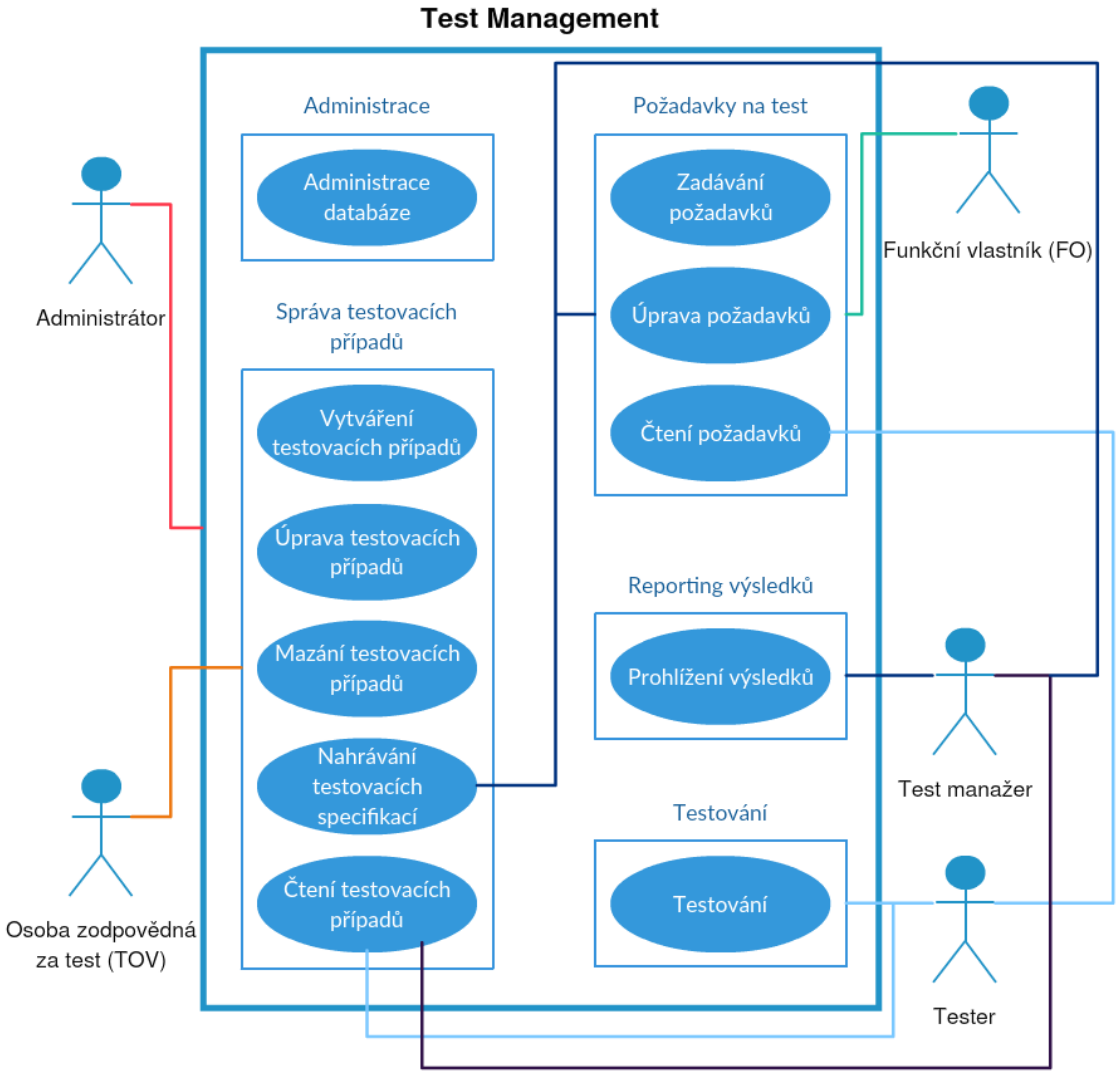
4.5 Role uživatelů a oprávnění

Přístup k aplikaci potřebují uživatelé s rozdílnými pracovními pozicemi. Různé skupiny uživatelů používají aplikaci za odlišným účelem. Aby byla zajištěna kontrola přístupu k jednotlivým částem aplikace a aby každý uživatel dostal obsah pro něj určený, navrhl jsem systém uživatelských rolí a k nim příslušných oprávnění.

Uživatel s nejvyššími právy je administrátor. Má přístup do všech částí aplikace, atribut *staff_status* mu zajišťuje i přístup do uživatelské administrace databáze. Další skupinou

uživatelů jsou osoby, které mají odpovědnost za testy – TOV (Test Objekt Verantwortlich). Ti mají oprávnění k používání části aplikace pro správu testovacích případů. Skupina TOV může provádět veškeré operace nad testovacími případy.

Skupina funkčních vlastníků (FO) má přístup do oblasti pro zadávání požadavků na test. Mají přehled o všech požadavcích ostatních uživatelů, vlastní požadavky mohou i zpětně editovat.



Obrázek 6: Uživatelské role a případy užití

Test manažer je oprávněn k prohlížení výsledků testů, může nahrávat existující testovací specifikace do databáze a také číst testovací případy. Test manažer má plný přístup do oblasti pro zadávání požadavků na test. Tester je kromě administrátora jediný uživatel s přístupem do testování. Může také číst požadavky na test a testovací případy. Každý uživatel může být součástí více skupin, díky tomu je možné existující oprávnění kombinovat a přizpůsobit systém momentálním potřebám.

5 Implementace

Aplikace je implementována v jazyce Python ve verzi 3.7.0 s použitím frameworku Django ve verzi 2.1. Pro uchovávání dat slouží relační databáze PostgreSQL ve verzi 10.5. Pro psaní zdrojového kódu jsem použil editor Atom.

Vytvoření Django projektu se provádí příkazem *startproject* [13]. Tento příkaz automaticky vygeneruje adresářovou strukturu a základní kód s nastavením projektu [14].

```
...\> django-admin startproject testcentrum_project
```

Samostatným funkčním celkům se v Django frameworku říká aplikace. V tomto případě je příkladem aplikace správa testovacích případů. Vytvoření aplikace se provádí pomocí utility *manage.py* v adresáři projektu.

```
...\> py manage.py startapp tcmanager
```

5.1 Adresářová struktura projektu

Kořenový adresář nese název projektu. Stejnojmenný podadresář je potom Python balíček pro daný projekt. Podadresář obsahuje soubor *wsgi.py*, který slouží jako vstupní bod pro webové servery kompatibilní s WSGI (komunikační protokol mezi webserverem a aplikací). Soubor *urls.py* slouží k dispečinku URL adres na úrovni projektu, jsou v něm uvedeny cesty k jednotlivým aplikacím. Dalším souborem projektu je *settings.py*, který popisuje veškeré konfigurace. Zbývající soubor v podadresáři se jmenuje *__init__.py*, tento soubor neobsahuje žádný kód, pouze identifikuje adresář jako Python balíček. Takto vypadá struktura projektu:

```
testcentrum_project/
    manage.py
    accounts/
    dashboard/
    planner/
    runner/
    stats/
    tcmanager/
    testcentrum_project/
        __init__.py
        settings.py
        urls.py
        wsgi.py
        static/
        templates/
```

V podadresáři *testcentrum_project* se také nachází adresáře *static* a *templates*. V adresáři static jsou uloženy CSS a JavaScript soubory. Adresář *templates* potom obsahuje soubor *base.html*, který je základem pro všechny ostatní HTML šablony, linkuje styly a skripty společné v rámci projektu a také definuje hlavní menu.

5.1.1 Konfigurace projektu

Všechna globální nastavení, která platí pro celý projekt se nachází v souboru *settings.py*. Během vývoje je zde třeba nechat proměnnou *DEBUG* nastavenou na hodnotu *True*. Díky tomu je pak možné při testování aplikace vidět v prohlížeči chybová hlášení. List s názvem *INSTALLED_APPS* obsahuje cestu ke všem aplikacím v projektu. Kromě vlastních aplikací jsem použil také aplikaci *admin* a její závislosti (z balíčku *django.contrib*), dále potom aplikaci *crispy_forms,* kterou jsem používal pro snazší kontrolu nad renderováním formulářů. Součástí nastavení projektu je také cesta ke kořenovému URL dispečeru, v tomto případě *testcentrum.urls*. Dalším důležitým nastavením je napojení na databázi ve slovníku *Databases*. Databázi jsem zaregistroval pod klíčem *default,* který je také slovníkem a obsahuje potřebné informace pro navázání spojení s databází – jméno databáze, databázový engine, uživatel, heslo, hosting, port).

5.1.2 Utilita manage.py

Součástí kořenového adresáře projektu jsou také podadresáře aplikací projektu a utilita *manage.py*, která umožňuje pomocí příkazového řádku jednoduchou interakci s Django projektem. Kromě dříve zmíněného příkazu *startapp* jsem během vývoje využíval tyto příkazy:

```
...\> py manage.py runserver
...\> py manage.py makemigrations
...\> py manage.py migrate
...\> py manage.py createsuperuser
...\> py manage.py collectstatic
```

Příkaz *runserver* spouští vývojový webový server, díky kterému je možné upravovat zdrojový kód aplikace a zároveň okamžitě sledovat změny v uživatelském rozhraní bez nutnosti restartovat server [15]. Django používá pro práci s daty Object-relational mapper (ORM), jedná se o knihovnu, která automatizuje přenášení dat z relační databáze do objektů programovacího jazyka a naopak [16]. Příkazem *makemigrations* se ukládají

změny, které byly provedeny v datovém modelu aplikace. Následně se příkazem *migrate* všechny tyto změny aplikují na relační databázi – synchronizuje se schéma databáze s datovým modelem aplikace.

Další uvedený příkaz *createsuperuser* vytvoří uživatele, který se může přihlásit do administrace projektu. Posledním příkazem je potom *collectstatic*, který nashromáždí veškeré statické soubory do jednoho adresáře.

5.1.3 Aplikace projektu

Projekt *testcentrum_project* jsem rozdělil do sedmi samostatných funkčních celků – aplikací (těmi jsou *admin*, *dashboard*, *planner*, *tcmanager*, *runner*, *stats* a *accounts*). Rozdělení neplatí jen z hlediska kódu, ale také z hlediska prezentace uživateli – co aplikace, to vlastní položka v liště hlavního menu. Jednotlivé aplikace podrobně představím později, veškerá data viditelná na doprovodných snímcích obrazovky z aplikace *Testcentrum* jsou fiktivní (reálná data firmy Škoda Auto jsou tajná). Každá z aplikací má předpřipravenou strukturu souborů a adresářů, která vznikne po provedení příkazu *startapp*. Těmi nejdůležitějšími jsou *urls.py*, *models.py*, *views.py* a adresář *templates*.

Soubor *models.py* slouží k definici datových objektů, podle kterých se utváří odpovídající databázové schéma. Díky ORM používám v celém projektu při práci s daty pouze Python objekty (žádné SQL příkazy). Jako příklad uspořádání aplikace uvádím adresářovou strukturu aplikace *tcmanager* (aplikace pro správu testovacích případů).

```
tcmanager/
    __init__.py
    admin.py
    apps.py
    filters.py
    forms.py
    models.py
    tests.py
    urls.py
    views.py
    templates/
    migrations/
```

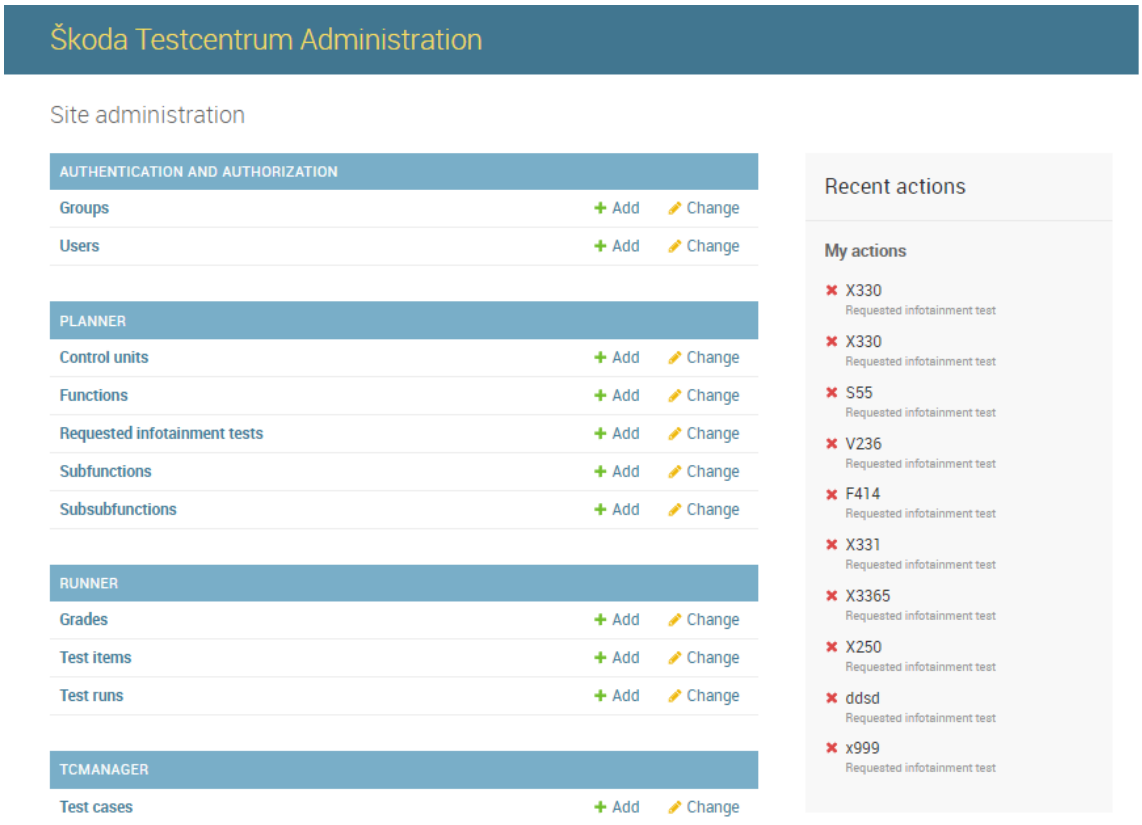
V Souboru *urls.py* jsou definovány všechny cesty v rámci dané aplikace. Ve *views.py* jsou funkce, které přijímají HTTP požadavky a vrací na ně odpověď, v tomto projektu obvykle HTML stránku. Nejčastěji se v těchto funkcích načítají a upravují objekty z databáze, které se pak ve slovníku *context* předávají do renderované HTML šablony. Zmíněné

funkce se volají na základě URL cesty, ke které jsou namapovány (přiřazení funkce k URL je definováno v *urls.py*).

V adresáři *templates* jsou uloženy všechny HTML šablony. Ty kromě statického HTML obsahují také dynamický obsah (předáván z *views.py* jako slovník). Pro vkládání dynamických dat do HTML slouží Django Template Language (DTL). Díky němu je možné generovat obsah stránek v cyklech, používat podmínky a další konstrukty.

5.2 Admin

Administrační rozhraní je automaticky generováno aplikací *admin* frameworku Django. Rozhraní je vytvářeno na základě datových modelů jednotlivých aplikací, datovými modely jsou myšleny třídy v souborech *models.py*. Aby se vytvořený model ukázal v administraci projektu, musí být zaregistrován v souboru *admin.py* v adresáři s aplikací příslušného modelu. Ve vygenerované administraci je možné jednoduše vkládat, upravovat a mazat záznamy do tabulek databáze pomocí formuláře. Rozhraní je možné customizovat – lze přidat filtrování záznamů, vlastní náhledy i hromadné operace se záznamy.



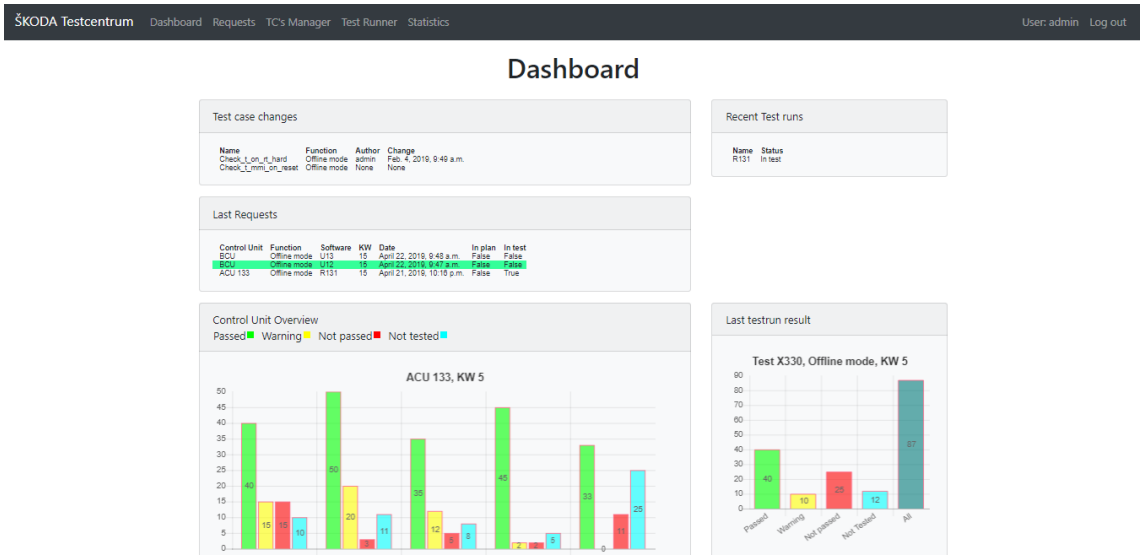
Obrázek 7: Administrace projektu

5.3 Dashboard

Pro odlišení termínu aplikace v projektu Django a pojmu aplikace ve významu celé test management aplikace, budu používat pro aplikaci jako celek název *Testcentrum*. Pro dílčí aplikace budu používat přímo jejich název ve zdrojovém kódu (vždy v nadpisu podkapitoly – v tomto případě *dashboard*).

Aplikaci *dashboard* jsem vytvořil v rámci projektu jako poslední. Jejím smyslem je poskytnout uživateli přehled o aktuálním dění v celém projektu, její obsah je tedy závislý na datech získaných ze všech ostatních aplikací. Jako první ji uvádím proto, že je pro uživatele vstupním bodem do aplikace *Testcentrum*.

V aplikaci *dashboard* se uživateli zobrazují poslední přidané požadavky na testování, nově přidané testovací případy, poslední vykonané testy a jejich výsledky v podobě grafů. Aplikace *dashboard* je přístupná pod kořenovou URL projektu. Po jejím zadání do prohlížeče se zavolá k ní přiřazená funkce, která jako vstupní parametr přijímá HTTP požadavek. V této funkci načítám několik posledních záznamů z databáze, které chci prezentovat uživateli, ty poté vkládám do slovníku *context*. Funkce vrací objekt *HttpResponse* s renderovaným textem, který získám pomocí funkce *render()* z balíčku *django.shortcuts*. Ta přebírá jako vstupní parametry HTTP požadavek, HTML šablonu a slovník s proměnnými nebo objekty, které chci v šabloně použít.

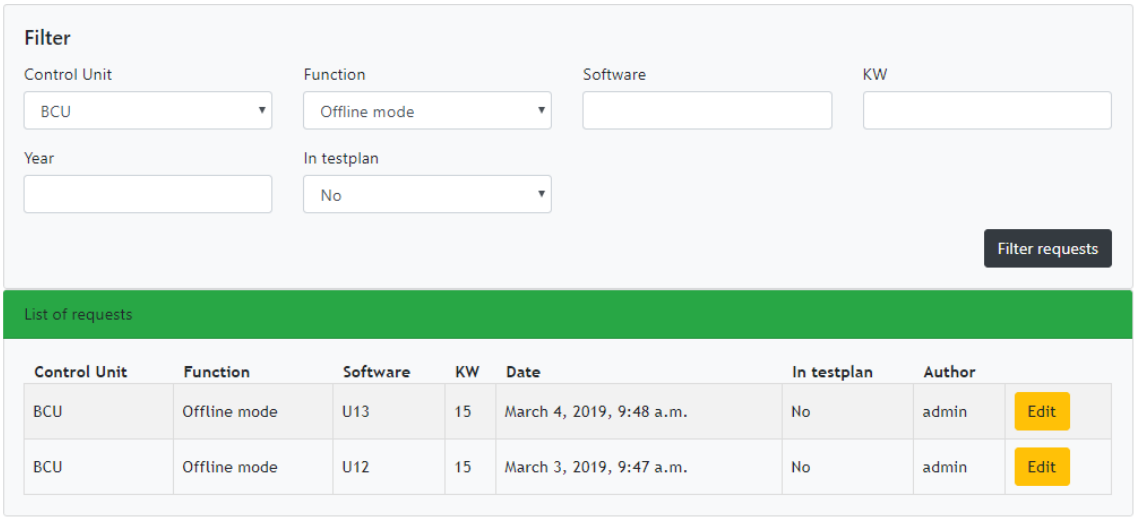


Obrázek 8: Dashboard

5.4 Planner

Primárním účelem aplikace *planner* je umožnit uživateli zadávat požadavky na test. Do aplikace se uživatel dostane přes položku hlavního menu – *Requests*. Na kořenové URL této aplikace je prezentována stránka se seznamem již zadaných požadavků, které je možné filtrovat.

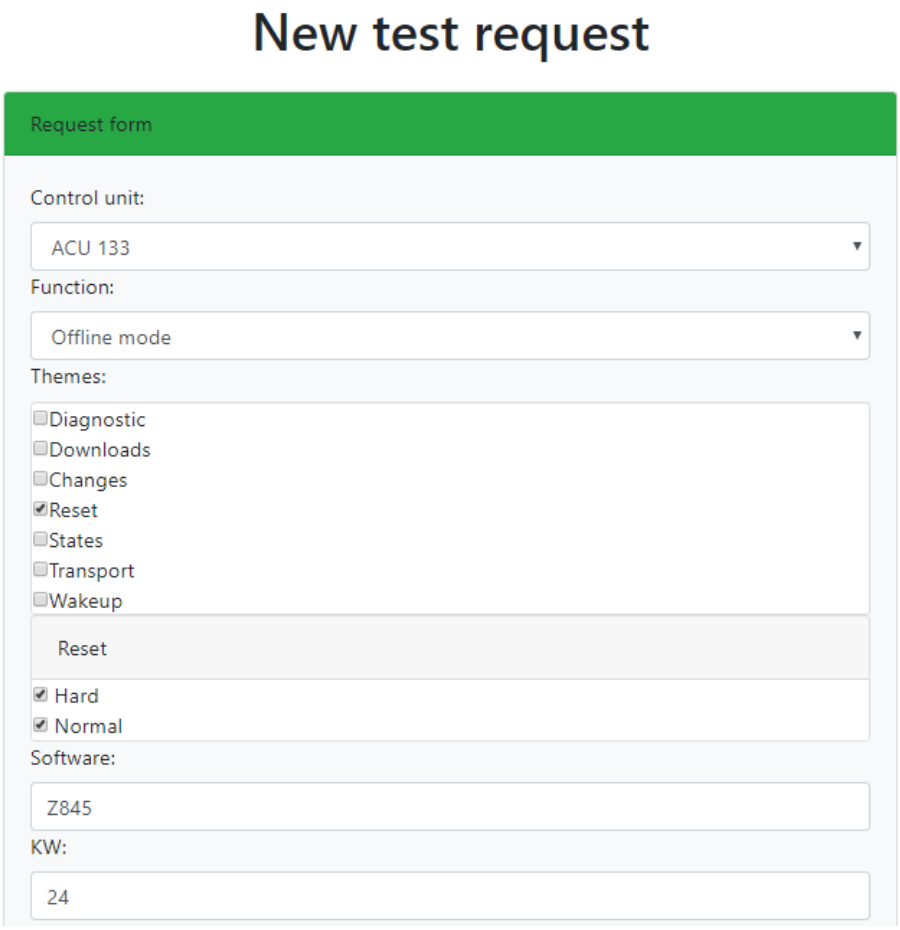
Pro filtrování požadavků jsem použil aplikaci *django_filter*, kterou jsem jednoduše nainstaloval pomocí nástroje pip. Do aplikace *planner* jsem poté přidal soubor *filters.py*, ve kterém jsem definoval třídu *RequestedTestFilter*. Ta dědí od třídy *FilterSet* z aplikace *django_filter*. Ve třídě *RequestedTestFilter* určuji model a konkrétní sloupce tabulky v databázi, které chci filtrovat. Ve *views.py* jsem definoval funkci, která je volána po odeslání formuláře filtru, zde vytvářím instanci třídy *RequestedTestFilter* a předávám ji objekt se záznamy k filtrování. Funkce vrací objekt *HttpResponse* s vyfiltrovanými daty.



Obrázek 9: Filtrování požadavků na test

K zadávání nových požadavků na test slouží dynamický webový formulář. Funkce se uživateli nabízí na základě toho, jakou zvolil v předchozím poli řídicí jednotku. Analogicky funkční témata a podtémata jsou nabízena podle zvolené funkce. Dynamický formulář je výrazným zlepšením této funkce oproti jejímu předchůdci v aplikaci Teamweb (kapitola 2.1.1). Zadávání testů je díky dynamicky nabízeným volbám přehlednější a rychlejší. K načítaní voleb do formuláře používám funkci *ajax* z knihovny jQuery, tato funkce provádí asynchronní HTTP požadavek. Díky tomu se volby mění bez nutnosti znovu načítat celou stránku.

V případě nabídky funkcí se *ajax* zavolá jako reakce na změnu řídicí jednotky. Z HTML elementu *option* získám hodnotu atributu *value*. Hodnotou je ID zvolené řídicí jednotky. ID asynchronně předávám Python funkci *load_functions*. Ta z databáze načte veškeré funkce patřící k vybrané řídicí jednotce a ty vloží do výběrového pole funkcí.



Obrázek 10: Ukázka formuláře pro zadávání požadavku na test

V seznamu požadavků na úvodní stránce aplikace *planner* lze zobrazit detail zvoleného požadavku. Požadavky lze odsud exportovat do souboru PDF nebo XLS. Export do XLS je možný i pro celý seznam požadavků. Pro export do formátu XLS používám funkci z knihovny jQuery table2excel. Export do formátu PDF je realizován ve vlastní funkci *renderPdf* s pomocí knihovny *xhtml2pdf*.

Požadavky v seznamu může oprávněný uživatel zpětně editovat. Každý uživatel může upravovat své vlastní požadavky (v seznamu lze vidět autora a čas změny) a uživatelé v roli test manažer mohou měnit stav požadavku podle toho, zda byl zapracován do test plánu, či nikoliv.

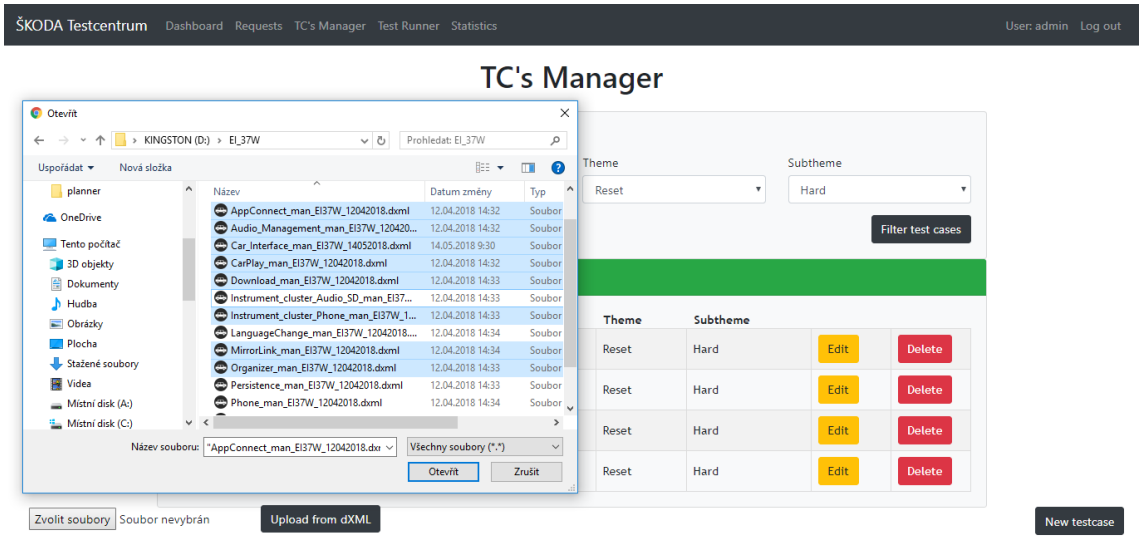
5.5 Tcmanager

Aplikace *tcmanager* je určena ke správě testovacích specifikací. Vytvářet, editovat a mazat testovací případy může uživatel v roli TOV. Test manažer může nahrávat do databáze testovací specifikace ve formátu DXML a číst testovací případy (přístup ke čtení má i uživatel v roli tester).

Seznam testovacích případů je možné filtrovat stejně jako v aplikaci *planner*. Tato funkce je implementována analogicky. Totéž platí pro vytváření testovacích případů, které je rovněž analogické s vytvářením požadavků na test v aplikaci *planner*.

Pro vkládání testovacích specifikací slouží funkce *load_test_specifications*. Uživatelem nahrané testovací specifikace jsou zde parsovány za použití Python modulu *xml.etree.ElementTree*. Z klíčových elementů DXML souborů získávám informace o jednotlivých testovacích případech a výsledná data následně vkládám do tabulky *tcmanager_testcase* v databázi. Nahrané testovací specifikace v DXML po přečtení neukládám.

K mazání a editaci testovacích případů používám *generic editing views* frameworku Django. Jedná se o třídy, které usnadňují implementaci opakujících se operací s formuláři. Editaci testovacího případu řeším ve třídě *TestCaseUpdateView*, která dědí od třídy *UpdateView*. Ve view definuji datový model – *TestCase* a přiřazuji třídu odpovídajícího formuláře – *TestCaseForm*. Nakonec určím HTML šablonu, do které se formulář vykreslí – *test_case.html*.

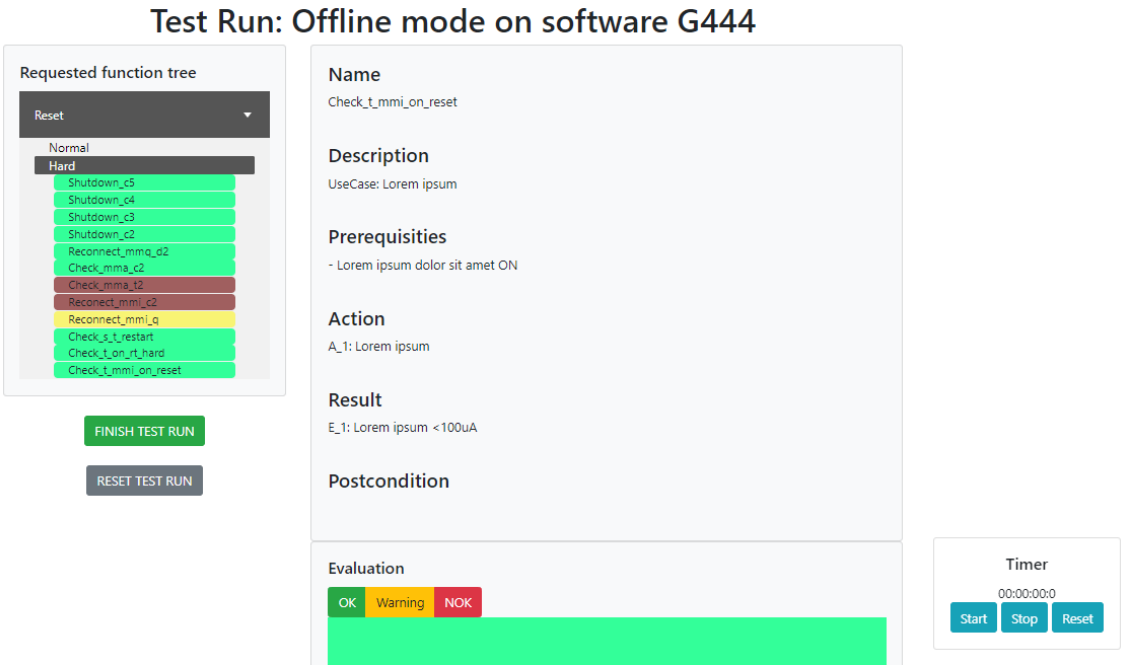


Obrázek 11: Nahrávání testovacích specifikací

5.6 Runner

Vykonávání testu řeší aplikace *runner*. Uživatel v roli tester nejdříve vyhledává požadavek, který má testovat, hledat může podle testované funkce nebo verze softwaru. Po nalezení odpovídajícího požadavku je možné začít test.

Testerovi se zobrazí rozbalovací seznam funkčních témat a podtémat, které byly zadány v požadavku na test. Tester může libovolně přepínat mezi tématy, po rozkliknutí tématu se rozbalí seznam podtémat. Rozbalení podtématu zobrazí seznam všech příslušných testovací případů. Po zvolení testovacího případu se uživateli načte jeho obsah, následuje vlastní zkouška a ohodnocení. Během testování má tester k dispozici manuální stopky.



Obrázek 12: Uživatelské rozhrání testování

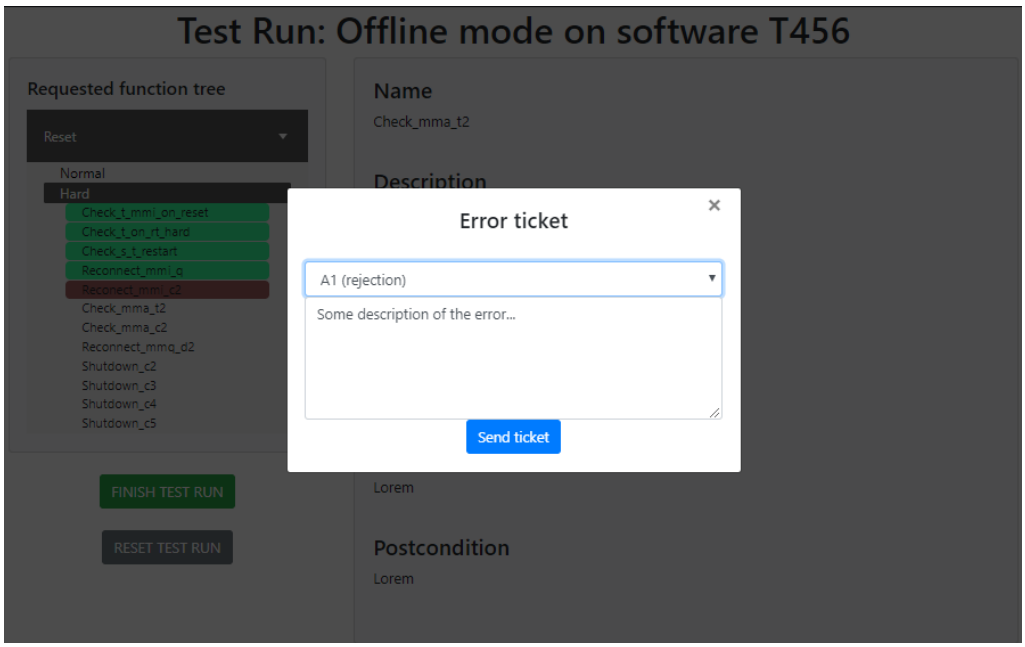
Proces vyhledávání požadavků řeší funkce *runner*. Ta na základě HTTP požadavku (metoda POST) provede dotaz na databázi a vrací stránku s výsledkem. V případě, že uživatel zahájí test, se zavolá funkce *testrun*. Zde se nastaví stav požadavku *in_test* na hodnotu *True*, následně se z databáze nahrají všechny požadované testovací případy. Pro každý testovací případ se vytvoří objekt *TestItem*, který slouží k uchování výsledku zkoušky testovacího případu (*passed, not passed, warning, not tested*). Pokud je výsledek negativní, tak se do objektu *TestItem* ukládá také klasifikace a popis chyby. Výchozí hodnota výsledku zkoušky (*result*) všech objektů *TestItem* je *not tested*.

Rozbalování seznamu s funkčními tématy jsem vyřešil pomocí JavaScriptu. Funkční témata jsem označil třídou *collapsible* (podtémata *collapsible2*) a na tuto třídu jsem nastavil *EventListener* na událost *click*. Při kliknutí uživatele na položku v rozbalovacím seznamu se nastaví vlastnost *display* třídy *innercontent* na *block*, v případě opětovného kliknutí se vlastnost nastaví na *none* a položka seznamu se skryje.

Po kliknutí na testovací případ se pomocí funkce *ajax* aktualizuje část stránky určená pro text testovacího případu. Přepínání mezi testovacími případy probíhá bez znovunačtení stránky a při ohodnocení testu aplikace automaticky předloží testerovi další testovací případ, který je v pořadí. Pro ukončení testu slouží funkce *testrun_finish*, požadavek na tento test se poté označí jako *finished* a dále se nenabízí v nabídce pro testování. Tato funkce zároveň shromáždí veškeré výsledky provedeného testu a předloží uživateli stránku se souhrnem testu. Test je také možné resetovat pomocí funkce *testrun_reset*.

Pokud je test vyhodnocen negativně zobrazí se formulář pro zadání chyby. Data formuláře jsou odesílány asynchronně pomocí funkce *ajax* metodou POST a ukládají se do příslušného objektu *TestItem*.

Stopky pro meření doby trvání některých akcí v testovacích případech jsem realizoval pomocí JavaScript knihovny EasyTimer.js. Vytvořil jsem novou instanci *Timer* a k ní jsem přiřadil *EventListenery*, které aktualizují zobrazovanou hodnotu času. K ovládání stopek používám funkce *start*, *pause*, *reset* a *stop*.

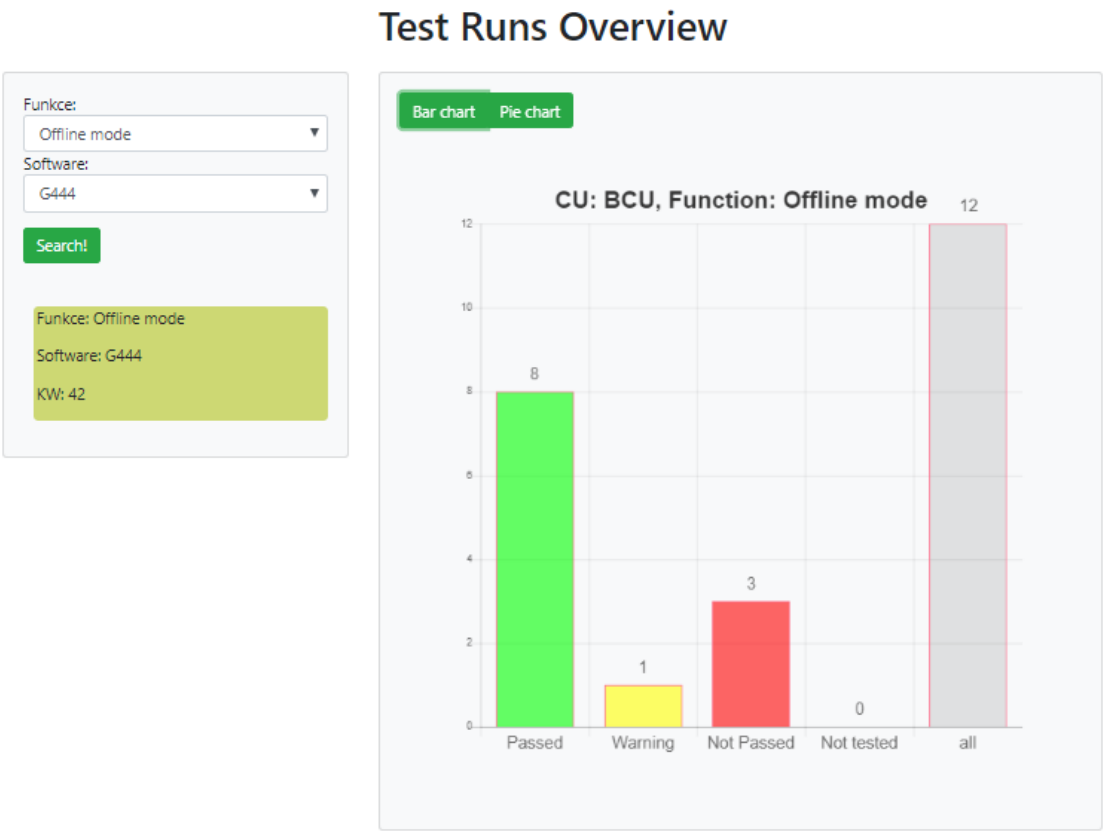


Obrázek 13: Formulář pro uložení chyby

5.7 Stats

Aplikace *stats* slouží k vizuální prezentaci testů. Výsledky je možné zobrazovat na úrovni jednotlivých testovaných funkcí nebo řídicích jednotek. Uživatel má k dispozici vyhledávání provedených testů (je možné zobrazovat i výsledky neukončených testů). Po zvolení testu se zobrazí sloupcový graf s výsledky.

Grafy vykresluji pomocí knihovny Chart.js do HTML elementu *canvas*, který je určen pro vykreslování grafiky do webové stránky. Stránku s grafem generuje asynchronně volaná funkce *load_testrun* ze souboru *views.py* aplikace *stats*. Tato funkce předává výsledky z databáze do šablony *graph.html*, zde je nalinkovaný JavaScript, který se stará o vykreslení grafu. Pro výchozí graf vytvářím instanci *Chart* s parametrem *barChartConfig*, kde je uložená konfigurace sloupcového grafu. Typ grafu je možné přepínat mezi sloupcovým a koláčovým grafem, k tomuto účelu jsem vytvořil funkci *changeChartType*, která vymaže předchozí instanci grafu a vytvoří instanci novou podle zvoleného typu (v případě koláčového grafu se vytvoří instance s parametrem *pieChartConfig*).

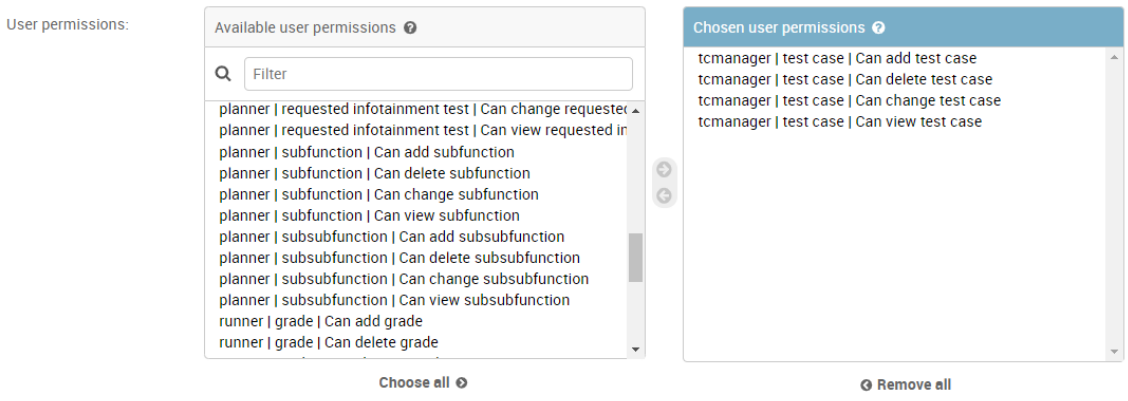


Obrázek 14: Zobrazení výsledku testu

5.8 Accounts

Operace s uživatelskými účty (přihlašování, odhlašování, změna hesla) je řešeno v aplikaci *accounts*. Aplikace využívá modul framworku Django *auth* z balíčku *django.contrib*. Tento modul zajišťuje jak autentizaci, tak autorizaci uživatele, zároveň je možná jeho customizace podle potřeb projektu. Použití modulu vyžaduje registraci modulů *auth* a *contenttypes* v listu *INSTALLED_APPS* v konfiguračním souboru *settings.py* (*content type system* umožňuje asociovat oprávnění s vlastními vytvořenými modely). Dále je nutné přidat do listu *MIDDLEWARE* položku *SessionMiddleware*, která spravuje *sessions* napříč HTTP požadavky, a *AuthenticationMiddleware*, který asociuje uživatele s HTTP požadavky prostřednictvím *sessions*.

Pro přihlášení uživatele jsem vytvořil funkci *login*, která přebírá jako vstupní parametr HTTP požadavek. Pro HTTP požadavek GET vrací server stránku s přihlašovacím formulářem. Pokud je metoda HTTP požadavku POST, volám funkci *auth.authenticate*, které předávám zadané uživatelské jméno a heslo z požadavku. V případě že jsou údaje platné, vrací funkce objekt uživatele. Ten je následně přihlášen pomocí funkce *auth.login* a přesměrován na kořenovou URL aplikace *dashboard*. Pokud údaje platné nejsou, server vrátí uživateli stránku s přihlašovacím formulářem a vypsanou chybovou hláškou. K odhlášení uživatele používám funkci *auth.logout*. Změna hesla se provádí funkcí *change*. Princip zpracování požadavku je analogický s přihlašováním uživatele. Uživatel nejprve zadá staré heslo a následně dvakrát po sobě heslo nové. Pokud je staré heslo platné a obě nová hesla jsou shodná, volám na objektu uživatele funkci *set_password*, které předávám nově zadané heslo. Následně objekt uložím metodou *save* a uživatele přesměruji do aplikace *dashboard*.



Obrázek 15: Přiřazení práv při vytvoření nebo změně uživatele

6 Testování a zpětná vazba

Aplikace byla testována průběžně během celého vývoje. Finální test probíhal v Testcentru elektroniky Škoda Auto od 4. dubna 2019 do 10. dubna 2019. Testování se zúčastnilo celkem 13 zaměstnanců na různých pracovních pozicích, které odpovídají některým rolím uživatelů v aplikaci. Konkrétně se jednalo o čtyři test manažery, čtyři testery, dva pracovníky se zodpovědností za funkci řídicí jednotky, specialistu na CAN komunikaci, koordinátora integračních testů a pracovníka zodpovědného za testovací místo.

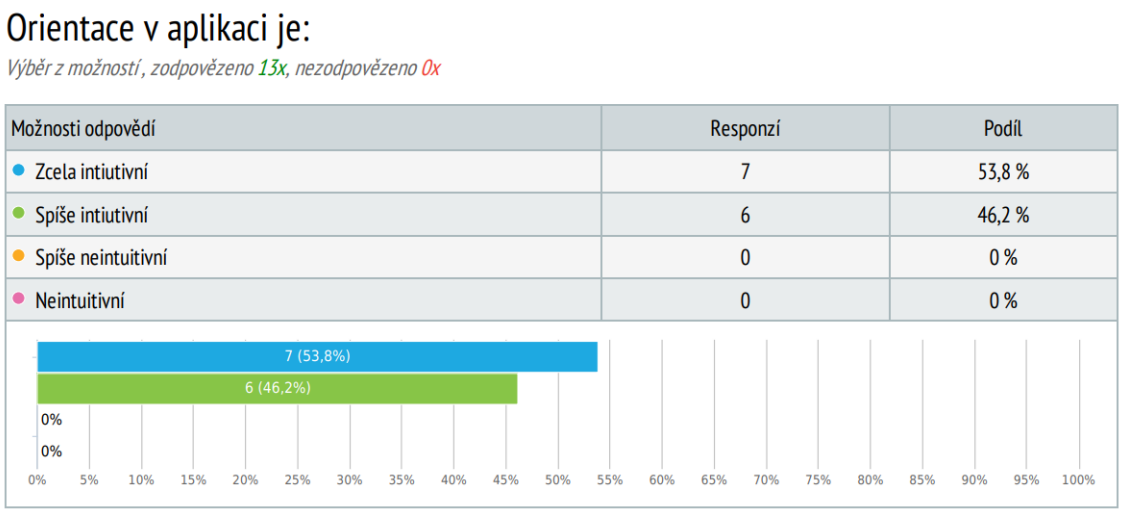
Účastníci byli seznámeni se všemi možnostmi aplikace a dostali obecný testovací scénář, který zaručoval vyzkoušení téměř všech funkcí systému. Při testování měli všichni možnost využít uživatele s přidělenými administrátorskými právy, aby měli přístup ke všem funkcím. Kromě připraveného scénáře, který zahrnoval celý proces od zadání požadavku až po sledování výsledků, byli účastníci vyzváni k testovaní vlastních scénářů dle svého uvážení.

Pro získání zpětné vazby jsem zvolil formu dotazníku. Ten dostali účastníci předem s pokynem k jeho vyplnění až po ukončení testování. Dotazník slouží k písemnému záznamu o nalezených chybách, získání hodnocení aplikace od zaměstnanců testcentra a shromáždění připomínek a návrhů na vylepšení aplikace. Znění celého dotazníku a kompletní výsledky je možné vidět v příloze A. Kromě získání zpětné vazby z dotazníku následoval po ukončení testu také osobní rozhovor s každým účastníkem testovaní. V tomto rozhovoru jsem zjišťoval podrobnější informace o nastalých chybách a jejich okolnostech. Testeři navíc detailněji sdělovali svoje poznatky z používání aplikace a přispěli mnoha připomínkami a možnými vylepšeními, které budu ještě zmiňovat později.

První otázka dotazníku byla určena k zaznamenaní pracovní pozice respondenta, profesní složení respondentů již bylo komentováno výše. Ve druhé otázce jsem zjišťoval, jaké testovací nástroje respondenti používají (mohli zvolit více možností). Tato informace je užitečná pro posouzení relevance hodnocení jednotlivých respondentů. Dvanáct ze třinácti respondentů používají alespoň jeden z nástrojů diskutovaných v kapitole 2 (Test management nástroje ve Škoda Auto). Ve třetí otázce jsem zjišťoval, zda respondenti preferují několik specializovaných nástrojů nebo by spíše dali přednost jednomu univerzálnímu nástroji. Devět ze čtrnácti respondentů dává přednost univerzálnímu nástroji.

Z Této informace se však vzhledem ke konečnému počtu účastníků testování nedají vyvozovat objektivní závěry.

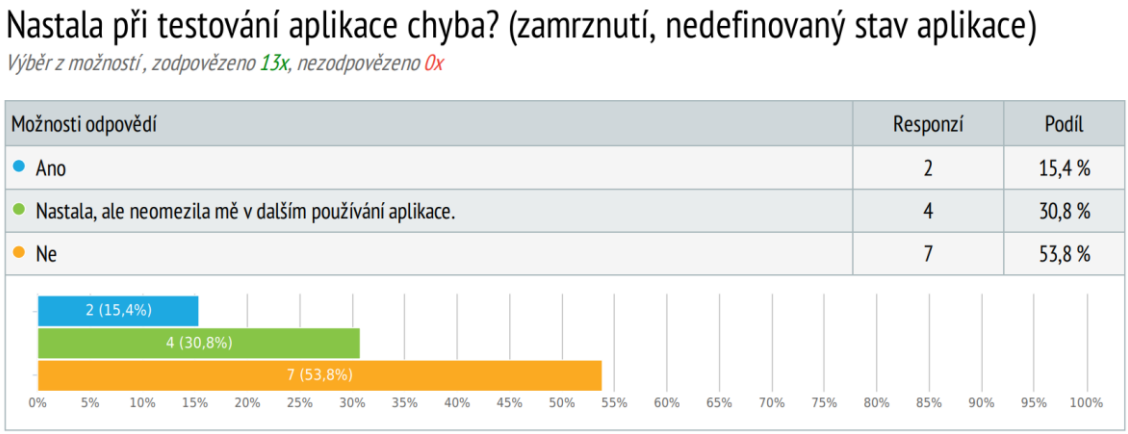
Další otázky jsou již věnovány zpětné vazbě k nové aplikaci. V otázce čtvrté hodnotili respondenti jak snadná nebo složitá pro ně byla orientace v aplikaci. Odpovědi jsou graficky znázorněny na obrázku níže (viz Obrázek 16).



Obrázek 16: Hodnocení orientace v aplikaci

V otázce páté hodnotili respondenti přívětivost uživatelského rozhraní na stupnici od jedné do deseti, přičemž deset je nejlepší hodnocení. Celkové průměrné hodnocení přívětivosti UI je 8,7.

Šestá otázka sloužila k zjištění, zda během testu došlo k chybě. Druh chyby jsem klasifikoval do dvou možností – chyba, která nebránila dalšímu používání aplikace a chyba, které vedla k tomu, že aplikace přestala pracovat (výsledek viz Obrázek 17).



Obrázek 17: Počet případů, kdy nastala nějaká chyba

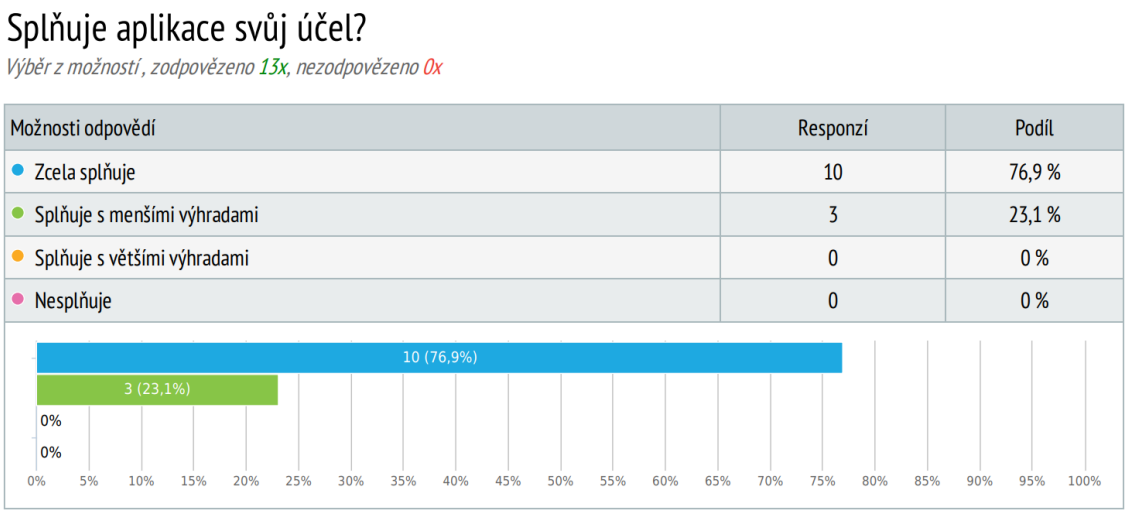
V otevřené sedmé otázce respondenti uváděli, na jakou chybu narazili. První zaznamenaná chyba se týká filtrování požadavků na test. Zde byl nedostatečně ošetřen vstup formuláře, při zadání záporného čísla do pole pro filtrování podle roku přidání požadavku aplikace skončila v chybovém stavu. Problém byl jednoduše vyřešen ověřením, zda je zadaná hodnota kladné číslo.

Další nalezenou chybou bylo nesprávné pořadí automatického přepínání testovacích případů v průběhu testu funkce řídicí jednotky. Po ohodnocení testovacího případu má být uživateli předložen automaticky následující testovací případ. Testovací případy však byly nesprávně indexovány a v některých situacích došlo k jejich přeskočení. Problém byl vyřešen pomocí funkce *order_by*, která byla použita k seřazení testovacích případů podle jejich ID.

Jeden z respondentů hlásil chybu, kdy se mu podařilo do formuláře pro zadání požadavku na test napsat vyšší hodnotu než 52 do pole pro kalendářní týden testu. Chyba byla opravena definováním minimální a maximální hodnoty daného formulářového pole.

Další chyba se týkala nežádoucího vícenásobného přidání testovacího případu. Při rychlém klikání na tlačítko *New testcase* docházelo k přidaní několika stejných záznamů testovacích případů do databáze. Chyba byla odstraněna znemožněním opětovného použití tlačítka po prvním stisknutí nastavením vlastnosti *disabled* na hodnotu *true*.

V otázce osmé respondenti odpovídali na otázku, zda aplikace jako celek splňuje svůj očekávaný účel. V následující otázce (deváté) pak měli uvést případné výhrady. Na obrázku níže jsou znázorněny odpovědi respondentů.



Obrázek 18: Odpovědi respondentů – otázka osmá v dotazníku

V desáté otázce hodnotili respondenti aplikaci jako celek na stupnici od jedné do deseti (deset je nejlepší hodnocení). Průměrné hodnocení aplikace jako celku je 9,1. V poslední otázce byli respondenti vyzváni ke sdělení svých návrhů na vylepšení aplikace. Zde se objevilo několik připomínek, které je dobré vzít v úvahu při dalším vývoji. Do všech návrhů respondentů je možné nahlédnout na poslední stránce přílohy A. Většina návrhů se týká drobných úprav uživatelského rozhrání. Jeden z návrhů se týkal přidání nové funkce do aplikace, konkrétně přidání porovnání více výsledků testů na jedné obrazovce.

Závěr

Provedl jsem rešerši současných test management nástrojů v Testcentru elektroniky Škoda Auto, ve které jsem upozornil na nedostatky některých řešení. Tyto nedostatky jsou důsledkem postupného nejednotného vývoje a nutnosti rychlého přizpůsobení se momentálním potřebám testování elektronických komponent nových vozů. Z rešerše vyplynulo, které nástroje je možné nahradit a jaké funkce je možné implementovat vhodnějším způsobem.

Před realizací vlastního řešení jsem provedl také rešerši existujících test management nástrojů na trhu. Dle žebříčků několika nezávislých zdrojů jsem vybral nejlépe hodnocené aplikace a posoudil jejich přínos pro testování elektroniky ve Škoda Auto. Žádná z nich se neukázala jako optimální řešení pro nasazení na daném pracovišti.

Na základě zjištěných poznatků v rešeršní části práce jsem navrhl vlastní webovou aplikaci pro účely test managementu. Určil jsem funkce, které bylo třeba realizovat a navrhl odpovídající databázi. Dle návrhu jsem aplikaci implementoval za použití webového frameworku Django v jazyce Python. Pro uchovávání dat byla použita databáze PostgreSQL. Nově vzniklá aplikace umožňuje uživatelům zadávat požadavky na testování, správu testovacích případů, manuální testování funkcí řídicích jednotek a sledování výsledků testů v grafické podobě.

Aplikace byla testována ve zkušebním provozu třinácti zaměstnanci Testcentra elektroniky. V průběhu testování bylo odhaleno několik nezávažných chyb, které byly následně opraveny. Zpětná vazba byla získána prostřednictvím dotazníkového šetření, jehož součástí jsou kromě hodnocení aplikace také návrhy na vylepšení. Deset respondentů uvedlo, že aplikace zcela splňuje svůj účel, zbylí tři uvedli, že aplikace splňuje svůj účel s mírnými výhradami.

V současném stavu je aplikace z technického hlediska připravena na použití v reálném provozu. Pro nasazení do produkce musí projít schvalovacím procesem a bezpečnostní certifikací. Do budoucna se nabízí implementovat nové návrhy zaměstnanců na pokročilejší funkce aplikace, zejména funkce pro přímé srovnání několika výsledků testů za různá období. Dalším možným rozšířením je správa testovacích míst a evidence řídicích jednotek.

Použitá literatura

[1] Computer Chips inside the Car. *CHIPS ETC.* [online]. [cit. 2019-03-13]. Dostupné z: http://www.chipsetc.com/computer-chips-inside-the-car.html

[2] Techopedia. In-Vehicle Infotainment (IVI). *Techopedia* [online]. [cit. 2019-03-13]. Dostupné z: https://www.techopedia.com/definition/27778/in-vehicle-infotainment-ivi

[3] Test Case. *Software Testing Fundamentals* [online]. [cit. 2019-03-15]. Dostupné z: http://softwaretestingfundamentals.com/test-case/

[4] Overview of Rational DOORS. *IBM* [online]. [cit. 2019-03-15]. Dostupné z: https://www.ibm.com/support/knowledgecenter/en/SSYQBZ_9.5.2/com.ibm.doors.requirements.doc/topics/c_welcome.html

[5] Top 20 Best Test Management Tools (New 2019 Rankings). *Software Testing Help* [online]. [cit. 2019-04-23]. Dostupné z: https://www.softwaretestinghelp.com/15-best-test-management-tools-for-software-testers/

[6] Best 25 Test Management Tools in 2019. *Guru99* [online]. [cit. 2019-03-22]. Dostupné z: https://www.guru99.com/top-20-test-management-tools.html

[7] RAJKUMAR. Best Test Management Tools of 2019. *Software Testing Material* [online]. 5. března 2019 [cit. 2019-03-22]. Dostupné z: https://www.softwaretestingmaterial.com/test-management-tools/

[8] DJANGO SOFTWARE FOUNDATION. The Django admin site. *Django* [online]. [cit. 2019-03-23]. Dostupné z: https://docs.djangoproject.com/en/2.2/ref/contrib/admin/

[9] Developer Survey Results 2019. *Stack overflow* [online]. [cit. 2019-04-20]. Dostupné z: https://insights.stackoverflow.com/survey/2019

[10] DJANGO SOFTWARE FOUNDATION. FAQ: General. *Django* [online]. [cit. 2019-03-20]. Dostupné z: https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names

[11] OS X Lion Server – Technical Specifications. *Apple* [online]. [cit. 2019-03-20]. Dostupné z: https://support.apple.com/kb/SP630?locale=cs_CZ

[12] ADAPTIC. Co je DOM | Adaptic. *Adaptic* [online]. [cit. 2019-03-20]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/dom/

[13] FORCIER, Jeff, Paul BISSEX a Wesley CHUN. *Python Web development with Django*. Upper Saddle River, NJ: Addison-Wesley, c2009. Developer's library. ISBN 978-0-13-235613-8.

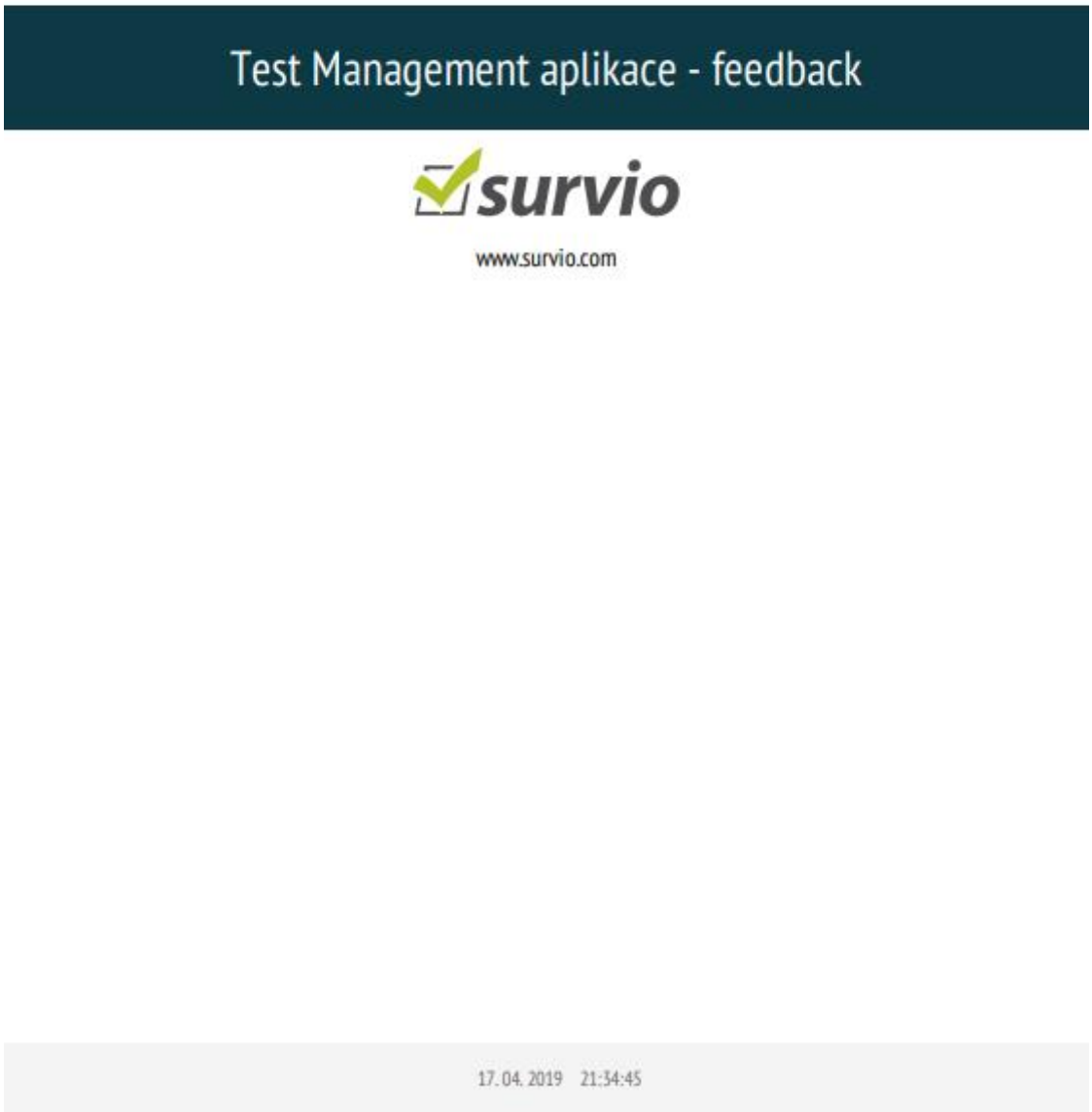
[14] DJANGO SOFTWARE FOUNDATION. Writing your first Django app, part 1. *Django* [online]. [cit. 2019-02-11]. Dostupné z: https://docs.djangoproject.com/en/2.2/intro/tutorial01/

[15] MELÉ, Antonio. *Django 2 by Example: Build powerful and reliable Python web applications from scratch*. Birmingham: Packt Publishing, 2018. Developer's library. ISBN 9781788472487.

[16] Object-relational mappers (ORMs). *Full Stack Python* [online]. [cit. 2019-02-11]. Dostupné z: https://www.fullstackpython.com/object-relational-mappers-orms.html

Přílohy

A Dotazník



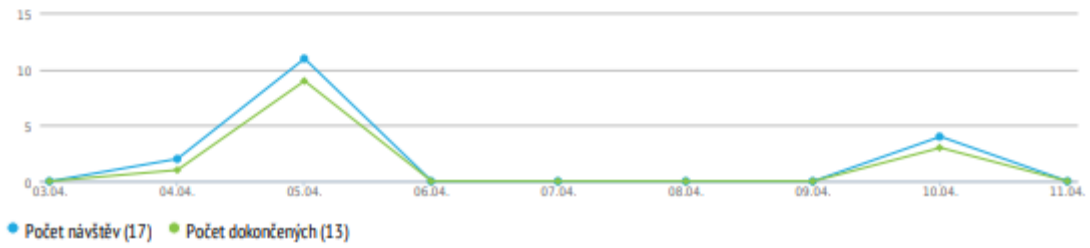
Základní údaje

Název výzkumu	Test Management aplikace - feedback
Autor	Vladimír Škopek
Jazyk dotazníku	Čeština
Veřejná adresa dotazníku	https://www.survio.com/survey/d/O2L1G9F6A7N8X3E8W
První odpověď	04. 04. 2019
Poslední odpověď	10. 04. 2019
Doba trvání	7 dnů

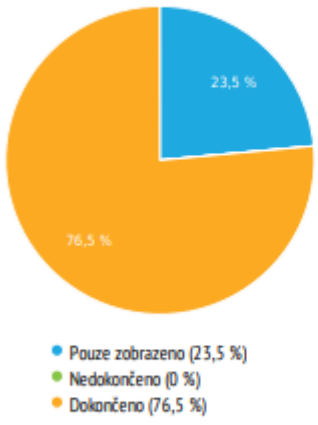
Statistika respondentů

17	13	0	4	76,5 %
Počet návštěv	Počet dokončených	Počet nedokončených	Pouze zobrazení	Celková úspěšnost vyplnění dotazníku

Historie návštěv (04. 04. 2019 – 10. 04. 2019)

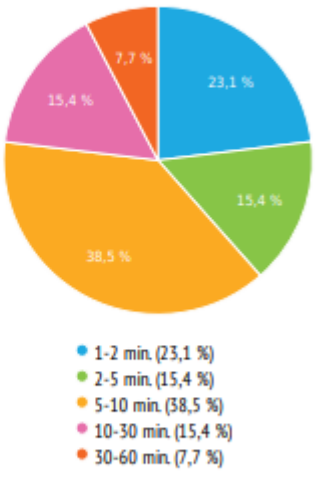


Celkem návštěv



Zdroje návštěv

Čas vyplňování dotazníku

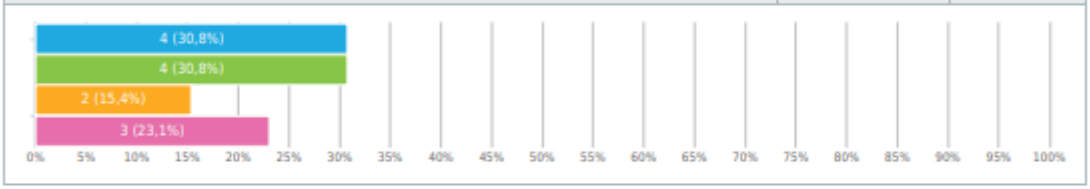


Výsledky

1. Pracuji jako:

Výběr z možností, zodpovězeno 13x, nezodpovězeno 0x

Možnosti odpovědi	Responzí	Podíl
Test manager	4	30,8 %
Tester	4	30,8 %
Osoba zodpovědná za funkci řídicí jednotky	2	15,4 %
Jiná...	3	23,1 %

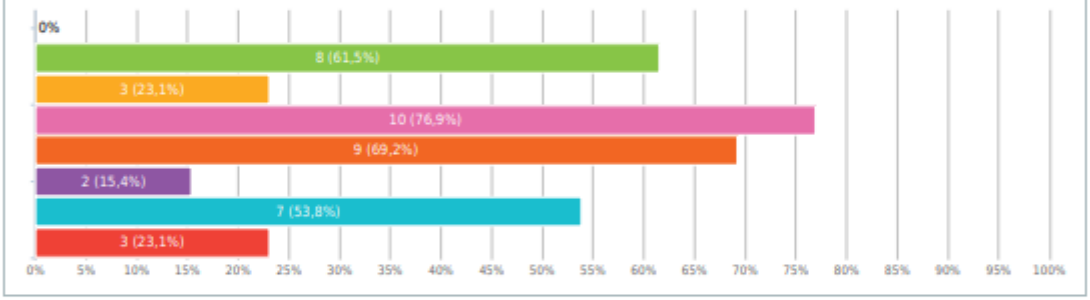


- specialista CAN
- koordinátor a "pate" Testcentrum integrační test Škoda
- Osoba zodpovědná za testovací místo

2. Které z následujících nástrojů využíváte?

Výběr z možností, více možných, zodpovězeno 13x, nezodpovězeno 0x

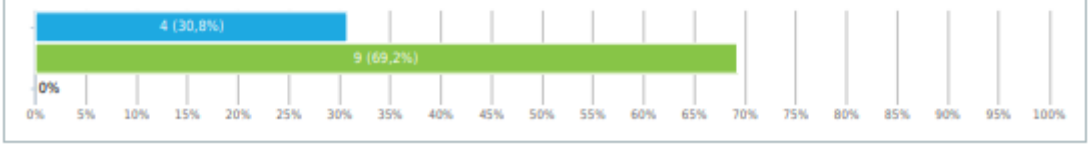
Možnosti odpovědí	Responzí	Podíl
Doors	0	0 %
Teamweb	8	61,5 %
dTCM	3	23,1 %
Teru	10	76,9 %
KPM	9	69,2 %
Das Cockpit	2	15,4 %
Sys42	7	53,8 %
Bussines Object	3	23,1 %



3. Dávám přednost:

Výběr z možností , zodpovězeno 13x, nezodpovězeno 0x

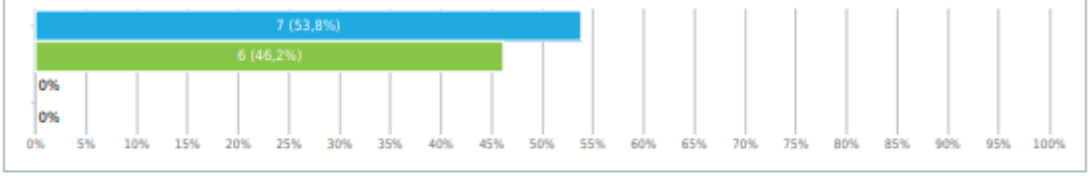
Možnosti odpovědí	Responzí	Podíl
většímu množství specializovaných aplikací	4	30,8 %
jedné aplikaci, zastupující většinu funkcí	9	69,2 %
Jiná...	0	0 %



4. Orientace v aplikaci je:

Výběr z možností , zodpovězeno 13x, nezodpovězeno 0x

Možnosti odpovědí	Responzí	Podíl
Zcela intiutivní	7	53,8 %
Spíše intiutivní	6	46,2 %
Spíše neintuitivní	0	0 %
Neintuitivní	0	0 %

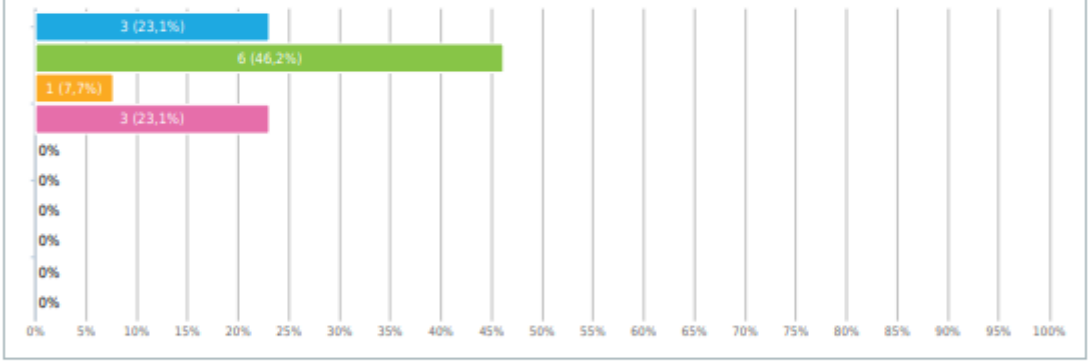


5. Ohodnoťte přívětivost uživatelského rozhraní:

Hvězdičkové hodnocení, zodpovězeno 13x, nezodpovězeno 0x

Počet hvězdiček 8,7/10

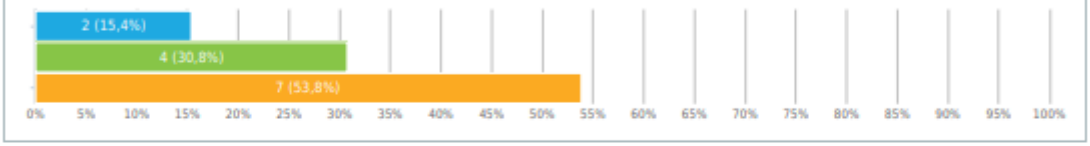
Odpověď	Responzi	Podíl
10/10 ★★★★★★★★★★	3	23,1 %
9/10 ★★★★★★★★★☆	6	46,2 %
8/10 ★★★★★★★★☆☆	1	7,7 %
7/10 ★★★★★★★☆☆☆	3	23,1 %
6/10 ★★★★★★☆☆☆☆	0	0 %
5/10 ★★★★★☆☆☆☆☆	0	0 %
4/10 ★★★★☆☆☆☆☆☆	0	0 %
3/10 ★★★☆☆☆☆☆☆☆	0	0 %
2/10 ★★☆☆☆☆☆☆☆☆	0	0 %
1/10 ★☆☆☆☆☆☆☆☆☆	0	0 %



6. Nastala při testování aplikace chyba? (zamrznutí, nedefinovaný stav aplikace)

Výběr z možností, zodpovězeno 13x, nezodpovězeno 0x

Možnosti odpovědí	Responzí	Podíl
Ano	2	15,4 %
Nastala, ale neomezila mě v dalším používání aplikace.	4	30,8 %
Ne	7	53,8 %



7. Pokud chyba nastala, popište ji

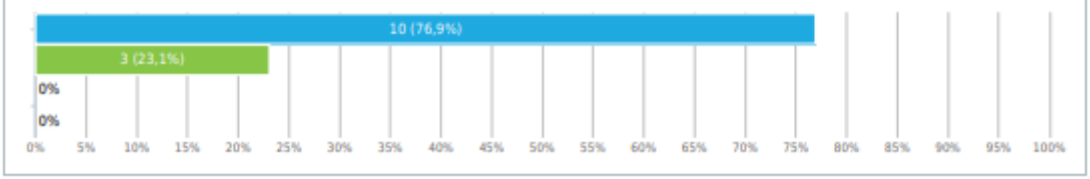
Textová odpověď, zodpovězeno 13x, nezodpovězeno 0x

- (8x) null
- při filtrování požadavku byla nalezena chyba
- Preskočenie testcasov
- Je možné zadat KW větší než 52.
- neošetřené tlačítko pro přidání TC -> důsledek vícenásobné přidání stejného TC
- Problém s indexováním jednotlivých testů při automatickém přeskakování na následující test

8. Splňuje aplikace svůj účel?

Výběr z možností, zodpovězeno 13x, nezodpovězeno 0x

Možnosti odpovědí	Responzí	Podíl
Zcela splňuje	10	76,9 %
Splňuje s menšími výhradami	3	23,1 %
Splňuje s většími výhradami	0	0 %
Nesplňuje	0	0 %



9. Uveďte výhrady

Textová odpověď, zodpovězeno 13x, nezodpovězeno 0x

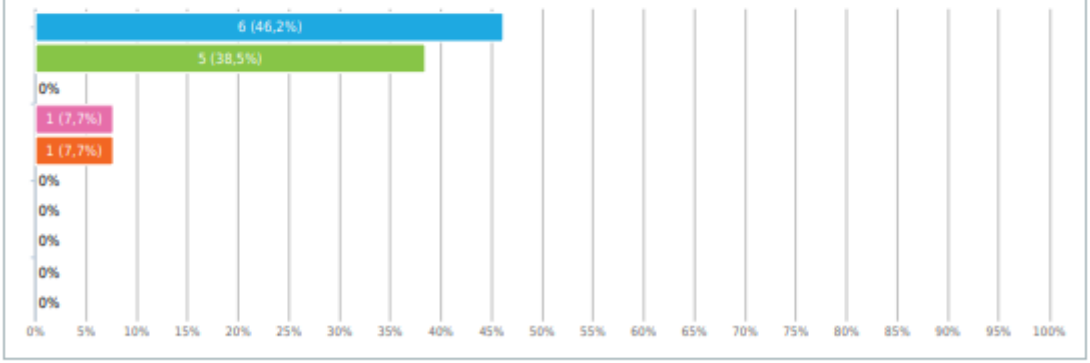
- (11x) null
- viz návrh
- Žádné nejsou

10. Vaše hodnocení aplikace:

Hvězdičkové hodnocení, zodpovězeno 13x, nezodpovězeno 0x

Počet hvězdiček 9,1/10

Odpověď	Responzi	Podíl
10/10 ★★★★★★★★★★	6	46,2 %
9/10 ★★★★★★★★★☆	5	38,5 %
8/10 ★★★★★★★★☆☆	0	0 %
7/10 ★★★★★★★☆☆☆	1	7,7 %
6/10 ★★★★★★☆☆☆☆	1	7,7 %
5/10 ★★★★★☆☆☆☆☆	0	0 %
4/10 ★★★★☆☆☆☆☆☆	0	0 %
3/10 ★★★☆☆☆☆☆☆☆	0	0 %
2/10 ★★☆☆☆☆☆☆☆☆	0	0 %
1/10 ★☆☆☆☆☆☆☆☆☆	0	0 %



11. Váš návrh na vylepšení aplikace (nebo libovolná poznámka):

Textová odpověď, zodpovězeno 13x, nezodpovězeno 0x

- (7x) null
- Hlavička exportu(KW_), Časová značka ve statistice, povinný komentář k nalezené chybě při testování,
- sladění do stejných barev pro přehlednost (OK-zelená...)
- výborný první krok k používání v rámci Testcentra. Důležité je nějaký čas tuto aplikaci používat, následně shromáždit doplňující požadavky, tyto roztřídit. a následně aplikovat.
- checkbox pro zapracování pouze pro testmanagera, reset/storno hodnocení TC, porovnání výsledků vedle sebe, více výsledků v jednom grafu (výběr více témat), provázanost skupiny testů dle výsledků
- Do kruhového grafu s výsledky bych nevkládal celkový počet testů. Při testování jednotlivých TCs by bylo dobré v seznamu vidět, který TC je otevřen/aktuálně prováděn
- Přidat možnost zrušit všechna označení pro vybírání testů, v přehledu testů poté přidat automatické rozbalování stromu testů

Nastavení dotazníku

Otázek na stránku	Všechny
Povolit odeslat vícekrát?	✓
Povolit návrat k předchozím otázkám?	✓
Zobrazovat čísla otázek?	✓
Náhodné pořadí otázek?	
Zobrazit ukazatel postupu?	✓
Oznámení o vyplnění dotazníku na e-mail?	
Ochrana heslem?	
IP omezení?	

Příloha: dotazník

Test Management aplikace - feedback

Tento dotazník slouží k získání zpětné vazby z testovaní aplikace pro test management ve Škoda Auto

1. Pracuji jako:

Nápověda k otázce: *Vyberte jednu odpověď*

- ○ Test manager
- ○ Tester
- ○ Osoba zodpovědná za funkci řídicí jednotky
- ○ Jiná... ☐

2. Které z následujících nástrojů využíváte?

Nápověda k otázce: *Vyberte jednu nebo více odpovědí*

- ☐ Doors
- ☐ Teamweb
- ☐ dTCM
- ☐ Teru
- ☐ KPM
- ☐ Das Cockpit
- ☐ Sys42
- ☐ Bussines Object

3. Dávám přednost:

Nápověda k otázce: *Vyberte jednu odpověď*

- ○ většímu množství specializovaných aplikací
- ○ jedné aplikaci, zastupující většinu funkcí
- ○ Jiná... ☐

4. Orientace v aplikaci je:

Nápověda k otázce: *Vyberte jednu odpověď*

- ○ Zcela intiutivní
- ○ Spíše intiutivní
- ○ Spíše neintuitivní
- ○ Neintuitivní

5. Ohodnoťte přívětivost uživatelského rozhraní:

Nápověda k otázce: *Více znamená lepší přívětivost UI.*

☆☆☆☆☆☆☆☆☆☆ ☐ / 10

6. Nastala při testování aplikace chyba? (zamrznutí, nedefinovaný stav aplikace)

Nápověda k otázce: *Vyberte jednu odpověď*

- ○ Ano
- ○ Nastala, ale neomezila mě v dalším používání aplikace.
- ○ Ne

7. Pokud chyba nastala, popište ji

8. Splňuje aplikace svůj účel?

Nápověda k otázce: *Vyberte jednu odpověď*

- ○ Zcela splňuje
- ○ Splňuje s menšími výhradami
- ○ Splňuje s většími výhradami
- ○ Nesplňuje

9. Uveďte výhrady

10. Vaše hodnocení aplikace:

Nápověda k otázce: *Čím více, tím lepší hodnocení.*

☆☆☆☆☆☆☆☆☆☆ ☐ / 10

11. Váš návrh na vylepšení aplikace (nebo libovolná poznámka):