

Projekt 1 - Testování a dynamická analýza 2023/24

- Autor, revize: Aleš Smrčka, 2024-03-13
- Název projektu: Návrh testů e-commerce systému OpenCart
- Přílohy: (žádné)

Cíl projektu

Cílem projektu je návrh testovací sady pro regresní testy webové aplikace:

1. Seznámit se s testovanou aplikací formou exploratorních testů (tj. seznámit se s aplikací “klikáním” v ní a používat vlastní zkušenosti). V sekci SUT níže je popsána část aplikace, na kterou by testy měly cílit.
2. Zvolit a navrhnout, co je cílem testů (které části povrchově do šířky nebo hloubkově do detailu budou testované).
3. Nastudovat chování řízený vývoj (Behaviour-driven development, BDD).
4. Vypracovat scénáře BDD pro testování zadané oblasti testované aplikace. Podrobné pokyny jsou níže.

Na 1. projekt navazuje 2. projekt, ve kterém budete implementovat vámi navržené testy pomocí nástroje Selenium a Behave [1]. Toto je třeba zohlednit při tvorbě testovacích scénářů BDD. Na hodnocení prvního projektu bude mít největší vliv váš výběr testovaných částí a kvalita vypracování testovacích scénářů. Není potřeba navrhnout komplikované testy nebo velký počet testů, ale efektivní testy, které s ohledem na svoji velikost pokryjí zajímavě velkou část aplikace.

Testovaná aplikace

e-Commerce systémy (pro internetové obchody) jsou velmi časté a mají mnoho implementací. Příkladem takového systému je open-source projekt [OpenCart](#). e-Commerce systémy umožňují jak samotné prohlížení produktů, nakupování, ale také vše, co s nimi souvisí v pozadí: správa objednávek, správa uživatelů, údržba skladovosti, správa produktů a variant, komunikace se zákazníky, analýzy, ...

Testovací cíle

Cílem projektu je **navrhnout** (nikoliv implementovat) regresní testy, které by měly zaručit, že po manipulaci webové aplikace (například její aktualizace) nevznikl problém. Tedy, že se nezmění hlavní funkcionality pro konzumenty webu.

Mezi aktivity webu, **které je třeba testovat**, patří:

1. Vyhledání a nákup zboží.
2. Správa zboží a jeho skladovosti.

Netestovaná funkcionality

Dalšími činnostmi ze seznamu níže **se nezabývejte**, pokud to nebude vyžadovat některý z testů.

1. Registrace a historie nákupů.
2. Správa objednávek.
3. Správa zákaznických účtů.

Nezabývejte se také dalšími činnostmi, které pestrá aplikace nabízí. V testech nezahrnujte činnosti pro nahrávání souborů, vizuální kontrolu obrázků a stahování souborů (například PDF faktur nebo štítků pro dopravu).

Pokyny k vypracování testovacích scénářů

Scénáře BDD vypracujte v souborech s příponou .feature. Syntaxi jazyka, význam příkazů a nejlepší praktiky pro psaní scénářů nastudujte dle dokumentace [2], doporučené četby a vlastních zdrojů. Minimální počet scénářů není stanoven. Zohledněte však jejich rozsah, protože ve druhém projektu budete implementovat odpovídající testovací případy pomocí automatizovaných testů GUI. Testování GUI bude tématem dalších přednášek.

Soubory se scénáři musí doprovázet report v souboru `README.md`, který bude obsahovat stručný popis testovaných vlastností **ve formě matice pokrytí**.

- Cílem souboru je seznámit čtenáře s rozsahem testovaných vlastností (dále jen *artefakty*) a vztahem těchto vlastností k jednotlivým testům a souborům popisující scénáře BDD.
- Soubor `README.md` bude ve formátu Markdown. Používejte co nejméně formátování, ideálně tedy variantu strict. Ověřit validitu textu ve formátu Markdown můžete na konvertoru [pandoc](#). Jazyk textu může být CZ/SK/EN, vše UTF-8.
- Matice pokrytí, které testy pokrývají které testované vlastnosti, bude ve formě 2 tabulek. První tabulka popisuje pokryté strukturální artefakty (webové stránky aplikace nebo ovládací prvky), druhá činnosti (aktivity, ověření vlastností). Sloupce v tabulkách jsou číselně označené testy. Pozn. obě tabulky musí mít tedy stejný počet sloupců.

Page	1	2	3	...
Page XYZ1	x	x		
Page XYZ2	x		x	
Page ...			x	x

Activities	1	2	3	...
Setting mapping OPQR to ABCD	x		x	
Deleting XYZ	x		x	
Doing something ...				x
Checking: relationAToB	x			x
Checking: relationXtoY		x	x	
Checking: A covers B	x	x	x	
Checking ...				x

- Matici pokrytí bude doplňovat matice implementovaných testů, tj. který test je pospán ve kterém souboru s BDD scénářem.

Feature file	1	2	3	...
file1.feature	x	x		
file2.feature			x	x

- Šablonu pro report `README.md` najdete v repozitáři s webovou aplikací: <https://pajda.fit.vutbr.cz/smrcka/its-2024/-/tree/master/template>

Způsob odevzdání

Soubory `.feature` a `README.md` zabalte do archivu `.zip` (bez vnitřní adresářové struktury). Archiv odevzdejte prostřednictvím informačního systému.

Hodnocení

Na výsledné hodnocení bude mít vliv počet scénářů, jejich délka v počtu kroků nebo unikátnost klazulí. Tyto metriky však samostatně nejsou dostatečné pro ohodnocení rozumné testovací sady. Hodnocení tedy zahrne také srozumitelnost testovacích scénářů a matice pokrytí.

Webová aplikace OpenCart

Testovanou aplikaci jako webovou aplikaci můžete rozjet na svém počítači pomocí PaaS technologie Docker (kontejnerová služba mimo jiné zajišťující také izolaci běžících aplikací). Pro tento účel je doporučen připravený obraz dostupný přes <https://hub.docker.com/r/bitnami/opencart/> (další návod včetně přihlašovacích údajů najdete na této stránce).

1. Nainstalujte si [docker](#) a [docker-compose](#).
2. Stáhněte si předpis pro běh kontejnerů potřebných pro OpenCart::

```
$ curl -sSL https://raw.githubusercontent.com/bitnami/containers/main/bitnami/opencart/docker-compose.
```

3. Spustíte kontejnery:

```
$ docker-compose up -d
```

Bez parametru `-d` uvidíte logy. První spuštění může trvat 10 s až 3 min.

4. Sledovat činnost aplikace lze pomocí logů:

```
$ docker-compose logs -f
```

5. Ve webovém prohlížeči je aplikace dostupná na adrese: `http://localhost/`

6. Administrativní rozhraní je dostupné na: `http://localhost/administration` Login a heslo je `user` a `bitnami`,

7. Pro ukončení běhu a opětovný start použijte:

```
$ docker-compose stop
```

```
$ docker-compose stop
```

8. Odstranění aplikace včetně dat lze pomocí:

```
$ docker-compose down -v
```

Pokud máte problém aplikaci zprovoznit lokálně, napište mi [email](#).

Literatura

- [1] Selenium a behave. <https://www.bddtesting.com/using-the-behave-framework-for-selenium-bdd-testing-a-tutorial/>
- [2] Syntaxe Gherkin pro psaní BDD. <https://cucumber.io/docs/gherkin/reference/>
- [3] Behaviour-Driven Development. <https://cucumber.io/docs/bdd/>
- [4] Nejlepší praktiky BDD. <https://automationpanda.com/2017/01/30/bdd-101-writing-good-gherkin/>
- [5] Antivzory BDD. <https://cucumber.io/blog/2016/07/01/cucumber-antipatterns-part-one>