Semestrální práce

Http request-response tester

Libor Matásek

Prohlašuji, že jsem práci vypracoval samostatně s využitím zdrojů uvedených ve zprávě.

1 Obsah

2 Konečná a schválená formulace zadání projektu

(Začíná na další straně.)

Specifikace projektu semestrální práce

Název projektu: Http request-response tester
Autor: Libor Matásek

1. Formulace řešeného problému

Webové stránky a aplikace se často příliš nezaobírají monitoringem služeb. Jaké překvapení, když se zjistí, že již půl roku od poslední úpravy nefunguje kontaktní formulář. Nebo když se o nefunkční API dozví od klienta.

Proto si aplikace klade za cíl být poměrně jednoduchým nástrojem pro testování webových stránek na bázi request-response. Aplikace poslouží např. pro otestování stránek po nasazení nových úprav. Definice testů jsou psané v XML, což je pro koncové uživatele (vývojáře) dobře známá technologie s výhodami jako je univerzálnost, čitelnost, možnost verzování, snadná přenositelnost a podpora v IDE díky schéma.

+ **Testy se ukládají síťově a je možno je sdílet dle definovaných práv.**

1.1 Cílový stav

Aplikace řeší problematiku testování webových stránek (formulářů, api, ...) implementací způsobů užití popsaných níže.

1.2 Způsoby užití

1.2.1 Nastavení

- Uživatel může v GUI nastavit složku pro ukládání http odpovědí z testů.
- Uživatel může nastavit složku s testy.

1.2.2 Definice testů

- Uživatel může vytvořit definici testu jako soubor xml umístěný v adresáři, který je nastaven v nastavení.

- Definice testu obsahuje: definici jednoho či více požadavků a definice testů na odpovědích těchto požadavků.

1.1.1 Testování odpovědi

Je možné testovat:

- Návratový HTTP status kód odpovědi.
- HTTP hlavičky odpovědi.
- Výskyt stringu v odpovědi.
- Zda odpověď splňuje časový limit.
- Zda je HTML validní (dle w3c validator).
- Zda HTML obsahuje pouze fungující odkazy.
- Existence klíče v JSON odpovědi (např. neobsahuje klíč „error:“ – vhodné např. pro jednoduché testování úspěšnosti odpovědí splňující protokol JSON-RPC).
- Existence tagu v XML odpovědi (např. neobsahuje element „<fault>“ – vhodné např. pro jednoduché testování úspěšnosti odpovědí splňující protokol XML-RPC).

1.1.2 Schéma testu

Je poskytnuto xsd schéma pro definici testu.

1.1.3 Seznam testů

Testy jsou zobrazeny v GUI ve stromovém zobrazení dle filesystemu.

1.1.4 Obnovení

Seznam lze z GUI obnovit, např. aby se objevil nově přidaný test. To má také funkci vyresetování GUI stavu aplikace.

1.1.5 Filtrování testů

Seznam testů je možné filtrovat vyhledávacím polem.

1.1.6 Spuštění testů

- Je možné spustit testy z GUI.
- Je možné spustit jen výběr testů označením složky či jednotlivých testů.

1.1.7 Průběh testů

Průběh testů je v GUI zobrazen progressbarem.

1.1.1 Přerušení spuštěných testů

Průběh testování je možné z GUI přerušit.

1.1.2 Úspěšné testy

Úspěšné testy jsou v GUI v seznamu testů barevně odlišeny.

1.1.3 Neúspěšné testy

Neúspěšné testy jsou v GUI v seznamu testů barevně odlišeny.

1.1.4 Detail dokončeného testu

- V seznamu testů lze vybrat dokončený test pro zobrazení detailu.
- V detailu jsou zobrazeny http odpovědi na jednotlivé requesty. Oznámení o případné chybě. Délka trvání testu.

1.1.5 Ukládání odpovědí

Http odpovědi v rámci testu jsou ukládány do filesystemu pro zobrazení v GUI. Uživatel si tak také případně může otevřít uložené odpovědi v jeho oblíbeném editoru pro hlubší prozkoumání.

1. Specifikace klíčových témat kurzu

1.1 Vláknové programování

Aplikace používá vláknové programování pro responsivní GUI. Jednotlivé testy jsou prováděny v samostatných vláknech.

1.2 Síťové programování

Aplikace používá síťové programování pro provedení požadavku na server a získání odpovědi.

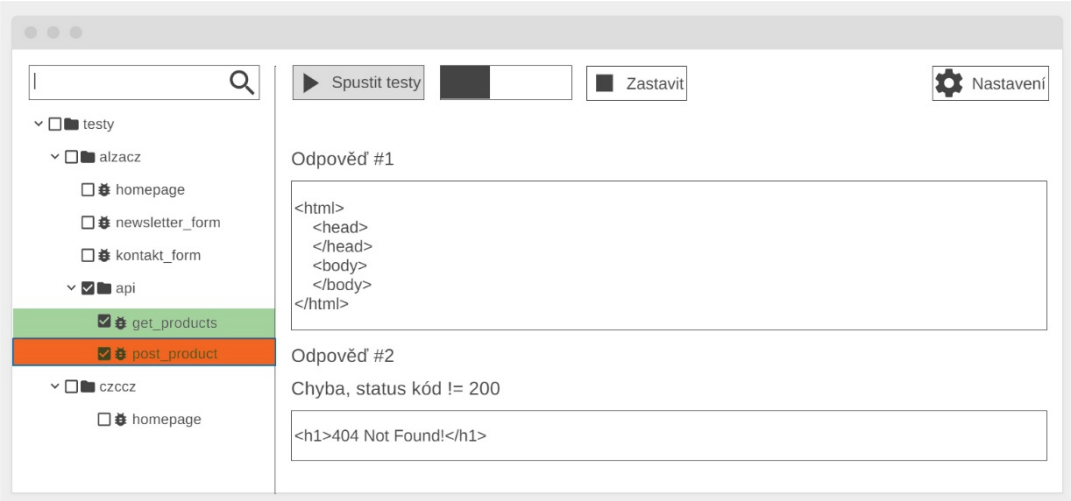
1.3 Práce s XML dokumenty

Aplikace používá XML dokumenty pro definice testů. Aplikace obsahuje XSD schéma definice požadavků. XML je použito také pro uchování nastavení.

1.Specifikace klíčových témat kurzu

1.1 Vizualizace grafického uživatelského rozhraní

1 Vizualizace hlavního GUI aplikace



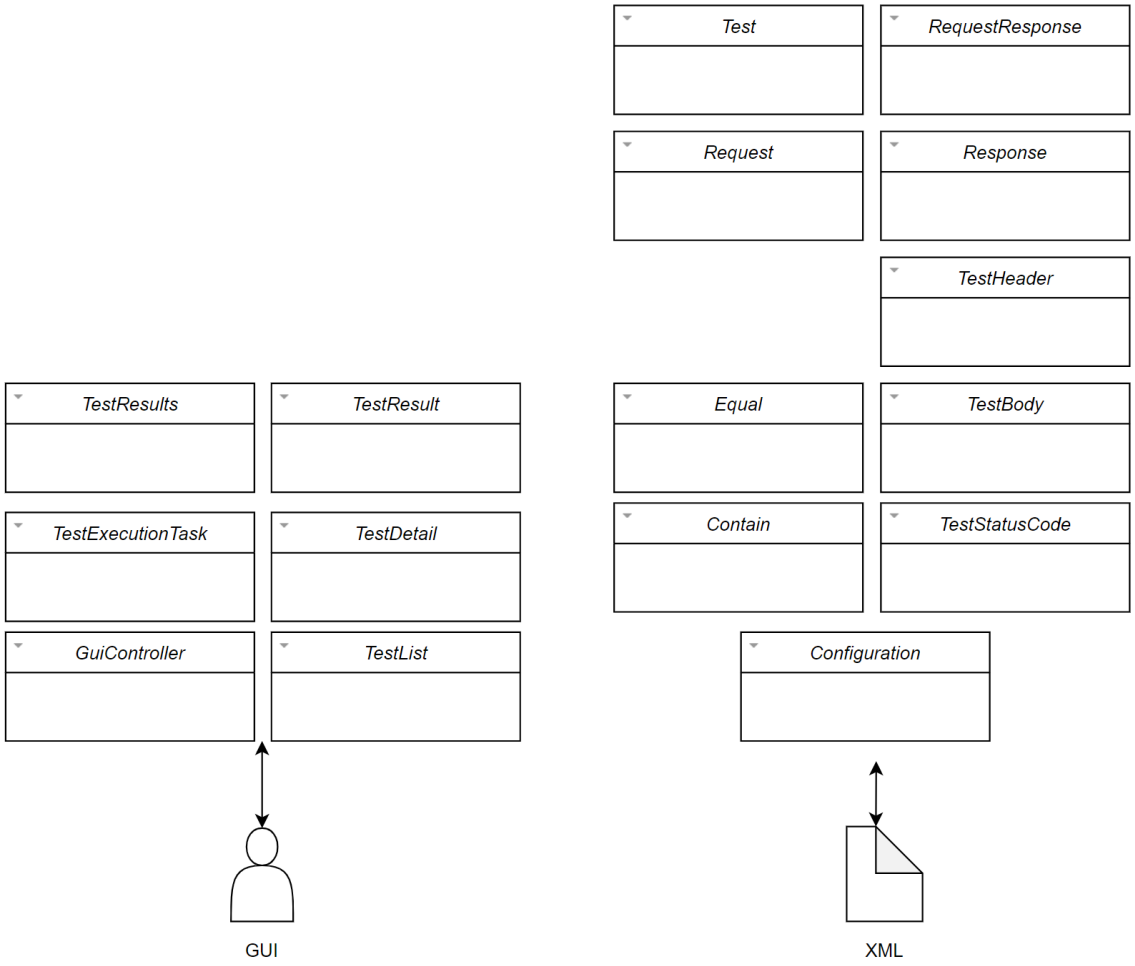
2 Vizualizace GUI nastavení

1.1 Návrh možné podoby testu

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<test
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="test.xsd"
>
  <request-response>
    <request>
      <url>http://www.example.com/api/newsletter</url>
      <method>POST</method>
      <header key="Content-Type">application/json</header>
      <body>
        {
          "email": "mail@example.com"
        }
      </body>
    </request>

    <response>
      <test-status-code>
        <equal>
          201
        </equal>
      </test-status-code>

      <test-body>
        <contain>
          success
        </contain>
      </test-body>

      <test-header key="Content-Type">
        <contain>
          application/json
        </contain>
      </test-header>
    </response>
  </request-response>
</test>
```

1.1 Návrh tříd

1 Zjednodušený model



- GuiController: hlavní GUI controller aplikace
- TestExecutionTask: task testu prováděný na pozadí v samostatném vláknu
- TestResults: akumulátor výsledků testů jednotlivých tasků
- TestResult: výsledek testu
- TestDetail: viewmodel
- TestList: viewmodel
- Configuration: konfigurační objekt aplikace
- Equal, Contain: predikáty testu
- TestHeader, TestBody, TestStatusCode: objekty pro testování různých komponent odpovědi

- Request: reprezentuje http požadavek
- Response: reprezentuje http odpověď
- Test: objekt definice testu
- RequestResponse: reprezentuje člen testu (jeden požadavek s odpovědí)

3 Analýza a popis funkcionality cílové aplikace na základě definovaných způsobů užití (scénářů)

3.1 Nastavení

Nastavení je ukládáno do xml souboru s názvem „app_settings.xml“ v kořenu projektu. Není ukládáno do databáze, neboť obsahuje cestu pro ukládání odpovědí. Pokud by se uživatel přihlásil na svůj účet z jiného stroje, cesta pro ukládání by nebyla validní, případně by si přepsal cestu pro původní stroj.

3.2 Definice testů

Definice testů jsou napsány v xml a ukládají se do databáze spolu s názvem testu, uživatelem databáze a právy pro přístup (čtení, čtení a úprava). Manipuluje se s nimi pomocí ORM, konkrétně jde o knihovnu LiteORM. Test je možno z plikace vytvořit, editovat i smazat. Definice testu neumožňuje více requestů v rámci jedné definice.

Přístup k testu je řešen na úrovni databáze i aplikace. Uživatel má přístup k testu:

1. Pro čtení a zápis: Je-li autorem testu, nebo má test nastaveno sdílení pro čtení a zápis.
2. Pro čtení: Je-li autorem testu, nebo má test nastaveno sdílení pro čtení či pro čtení a zápis.

Na úrovni databáze jsou oprávnění řešena takto:

```
-- Opravneni

-- Vytvoreni skupiny pro uzivatele aplikace
CREATE
GROUP app_users;
-- Prirazeni vsech opravneni na tabulce testy skupine uzivatelu aplikace
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE test TO GROUP app_users;
-- Prirazeni opravneni na generatoru id skupine uzivatelu aplikace
GRANT USAGE, SELECT ON SEQUENCE test_id_seq TO app_users;

-- Zapnuti funkce ROW LEVEL SECURITY - opravneni (policy) pro jednotlive radky
ALTER TABLE test ENABLE ROW LEVEL SECURITY;

-- Vsichni mohou insertovat
CREATE
POLICY insert_all ON test
    FOR
INSERT WITH CHECK (true);

-- Author testu muze na svem zaznamu delat vse
CREATE
POLICY author_all ON test
    USING (test.dbuser = CURRENT_USER);

-- Ostatni mohou zaznam cist pokud je sdilen ke cteni
CREATE
POLICY other_read_if_shared_for_read ON test
    FOR
SELECT
    USING
    (test.shared_for_read = true);

-- Ostatni mohou zaznam upravit a cist (update pouziva select) pokud je sdilen k uprave
CREATE
POLICY other_update_if_shared_for_update ON test
    USING (test.shared_for_update = true);
```

Obrázek 1 řešení oprávnění přístupu k testu v DB

Nového uživatele stačí přiřadit do skupiny „app_users“.

3.3 Testování odpovědi

Jsou implementovány všechny testy ze specifikace. K většině je napsán i test. Některé testy, kde to dává smysl, používají k rozhodnutí o (ne)správné hodnotě třídy predikátů. Číselné jako je rovnost, menší než, v intervalu a řetězcové jako je rovnost a obsahuje. Ke všem predikátům je napsán test. Množina predikátů, které je v testu možno použít, je specifikována v JAXB anotacích.

3.4 Schéma testu

Schéma bylo vygenerováno pomocí JAXB a je umístěno v „schemas/schema1.xsd“. Generování schema je nastaveno jako plugin mavenu v souboru „pom.xml“, spouští se tedy jako Maven goal – JAXB Schemagen.

3.5 Seznam testů

Testy jsou zobrazeny v GUI ve stromovém zobrazení. Nachází se zde kořen s testy přihlášeného uživatele a kořen pro sdílené testy dál dělený dle uživatelů.

3.6 Obnovení

Tlačítko obnovit resetuje test který je uložen v řádku stromu a výsledek posledního průběhu testu. Je ho tedy třeba použít po přidání, či úpravě testu aby se změny projevily. Také resetuje styly ve stromu.

3.7 Filtrování testů

Testy lze vyhledávat dle vstupu pro vyhledání. Odpovídající testy jsou zvýrazněny. Není tedy implementováno přímo filtrování.

3.8 Spuštění testů

Testy je možné spustit pomocí tlačítka „Spustit“. Provádějí se v ThreaPoolu v samostatných vláknech pro efektivitu a zachování responzivity GUI. Je možné vybírat konkrétní testy či celé skupiny.

3.9 Přerušení spuštěných testů

Testy je možné přerušit. Přerušení je zavoláno na příslušném ThreadPoolu.

3.10 Úspěšné testy

Úspěšné testy jsou v seznamu testů barevně rozlišeny. Styl řádku je nabindován na výsledek posledního testu.

3.11 Neúspěšné testy

Neúspěšné testy jsou v seznamu testů barevně rozlišeny. Styl řádku je nabindován na výsledek posledního testu.

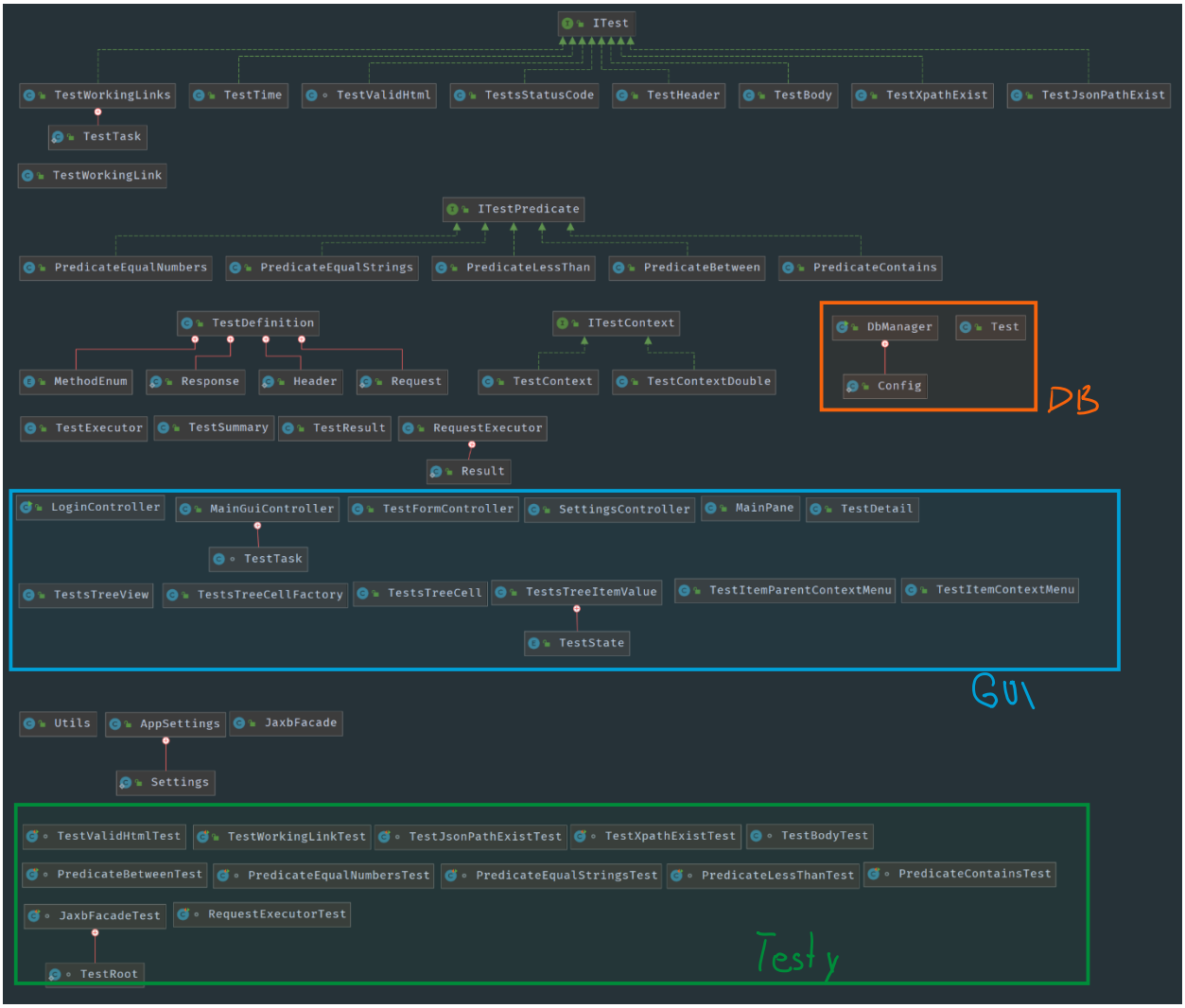
3.12 Detail dokončeného testu

V seznamu testů lze vybrat dokončený test k zobrazení detailu. V detailu jsou zobrazeny délky trvání samotného testu a requestu. Dále jsou zde zobrazeny úspešné a neúspešné testové definice spolu s dalšími informacemi, jako jsou např. testovaná hodnota.

3.13 Ukládání odpovědí

Je možné vypnout a zapnout v nastavení. Dále je možné nastavit cestu pro ukládání. Název souboru je složen z uživatele, názvu testu a časového razítka.

4 Diagram tříd projektu s jejich popisem a uvedením zodpovědností jednotlivých tříd



- Testový aparát:
  - ITest: Interface jednotlivých testů response.
  - TestWorkingLinks: Testuje zda jsou v response pouze fungující odkazy. Parsuje odkazy z html a volá TestWorkingLink.
  - TestWorkingLink: Testuje zda odkaz funguje.
  - TestTime: Testuje zda přijde response ze serveru včas.
  - TestValidHtml: Testuje zda je html odpověd validní dle w3c.
  - TestStatusCode: Testuje status code response.
  - TestHeader: Testuje http hlavičky response.

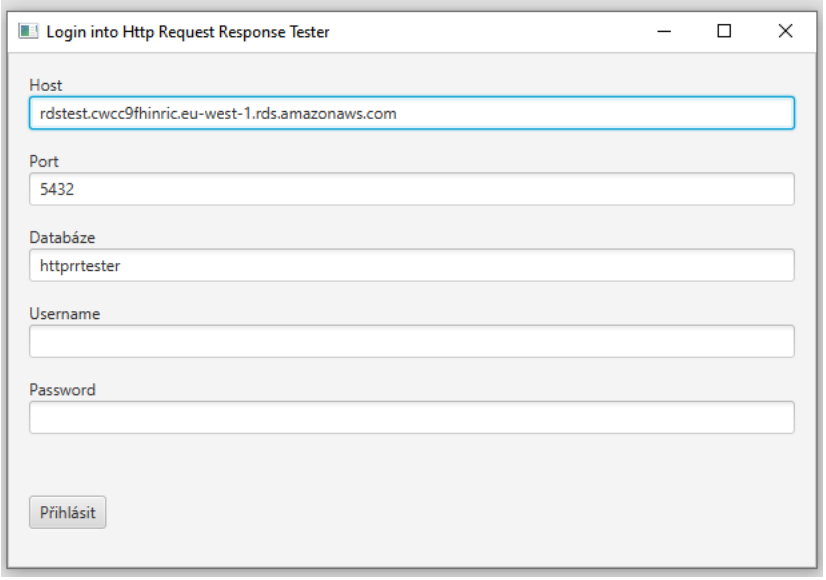
    - TestBody: Testuje tělo response.
    - TestXpathExist: Testuje existenci Xpath (xml tagu).
    - TestJsonPathExist: Testuje existenci JSON Path (json klíče).
    - ITestPredicate: Interface jednotlivých predikátů. Predikát je objekt, který je používán testem pro určení, zda je hodnota správná/špatná.
    - PredicateEqualNumbers: Predikát zda jsou čísla shodná.
    - PredicateEqualStrings: Predikát zda jsou řetězce shodné.
    - PredicateLessThan: Predikát zda je číslo menší než.
    - PredicateBetween: Predikát zda je číslo v intervalu.
    - PredicateContains: Predikát zda řetězec obsahuje podřetězec.
    - TestDefinition: Mapování a objektová reprezentace xml test definice.
    - ITestContext: Interface test contextu. Kontext se předáva testům a je to datová struktura obsahující data potřebná pro testy.
    - TestContext: Implementace ITestContext.
- Databáze:
    - DbManager: Singleton přístup k databázi.
    - DbManager.Config: konfigurace DB managera (např. connection string).
    - Test: Databázový model testu, ORM mapování.
- Gui:
    - LoginController: Vstupní bod aplikace, přihlášení k databázi.
    - MainGuiController: Controller hlavního GUI aplikace.
    - MainGuiController.TestTask: Task zahájení testu prováděný ve vlastním vláknu mimo JavaFX Application Thread.
    - TestFormController: Controller formuláře testu (nový, úprava).
    - SettingsController: Controller nastavení aplikace.
    - MainPane: Custom control hlavní panel – urvnitř např. detail provedeného testu.
    - TestDetail: Custom control detail (výsledek) proběhnutého testu.
    - TestsTreeView: Custom control stromu testů.
    - TestsTreeCellFactory: Továrna buněk stromu testů.
    - TestsTreeCell: Custom control jedné buňky v stromu testů.
    - TestsTreeItemValue: Hodnota buňky v stromu testů.

    - TestItemParentContextMenu: Context menu při kliku na rodičovskou buňku.
    - TestItemContextMenu: Context menu při kliku na buňku testu (listovou).
    - Utils: Utility aplikace.
    - AppSettings: Přístup k nastavení aplikace.
    - AppSettings.Settings: Xml definice nastavení aplikace.
    - JaxbFacade: Utility a zjednodušené volání JAXB.
- Testy – Vypsány pouze názvy, jejich zodpodvědností je testování příslušné třídy.
    - TestValidHtmlTest
    - TestWorkingLinkTest
    - TestJsonPathExistTest
    - TestXpathExistTest
    - TestBodyTest
    - PredicateBetweenTest
    - PredicateEqualNumbersTest
    - PredicateEqualStringsTest
    - PredicateLessThanTest
    - PredicateContainsTest
    - JaxbFacadeTest
    - RequestExecutorTest
    - TestContextDouble: Test double implementace ITestContext.

5 Popis návrhu grafického rozhraní a screenshoty jeho podstatných částí

5.1 Okno přihlášení do aplikace

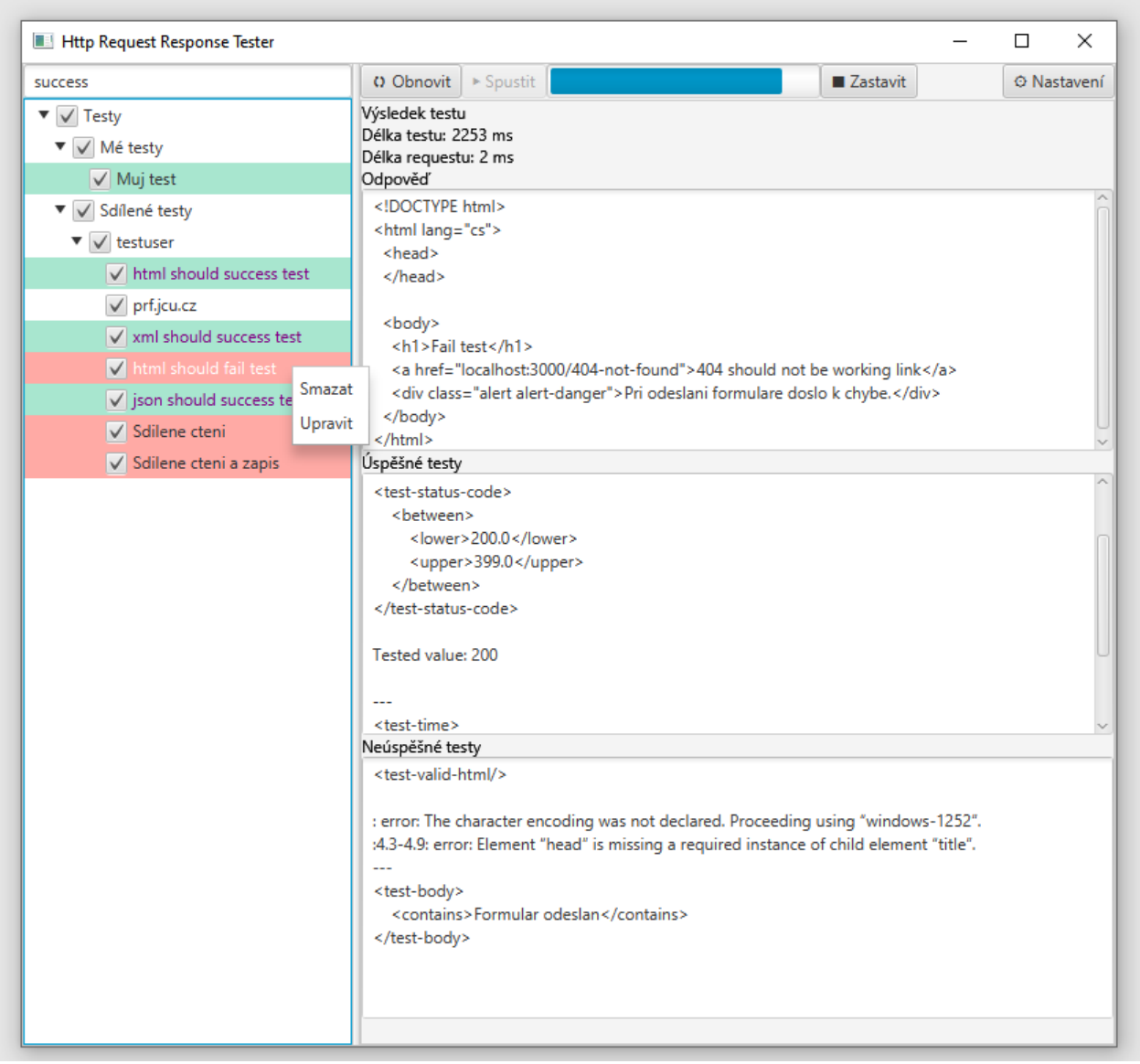
Vstupní bod aplikace. Umožňuje zadat přihlašovací údaje pro připojení k databázi organizace.



Obrázek 2 přihlašovací okno

5.2 Hlavní okno aplikace

Hlavní okno aplikace umožňuje především vybrat, provést a zobrazit výsledky testů. Hlavní komponenty jsou strom testů, hlavní panel a panel akcí. Okno je vertikálně rozděleno na dvě části SplitPanelem a je tak možné měnit proporce mezi levou a pravou částí.



Obrázek 3 hlavní okno aplikace

Komponenty:

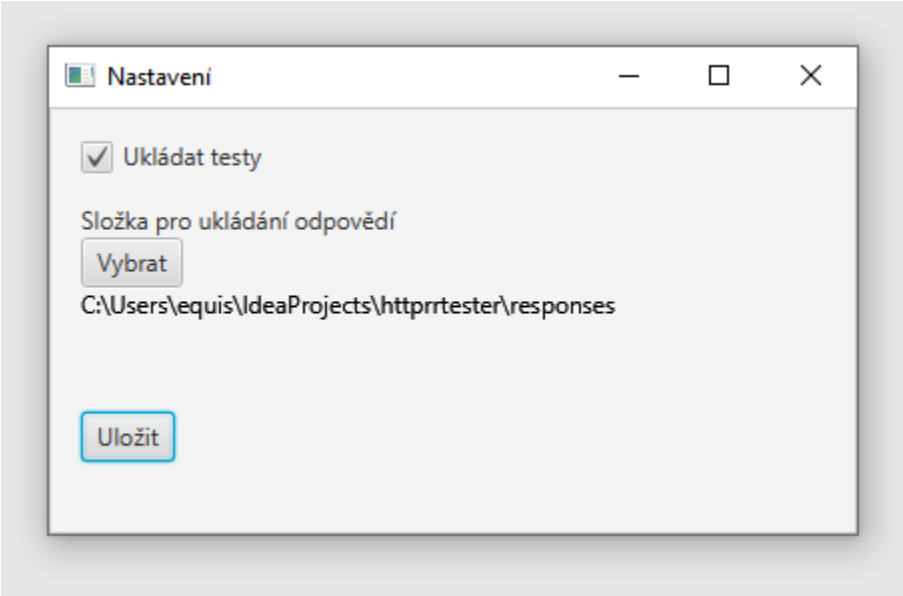
- Strom testů: Hierarchický výpis testů. Nachází se zde testy uživatele a sdílené testy ostatních uživatelů. Je možné vybrat pro testování konkrétní test nebo rodiče a jeho děti označením checkboxu. Testy je možné vyhledat pomocí textového pole, což odpovídající testy zvýrazní.

Pozadí itemů jsou při průběhu testů zvýrazněny dle výsledků. Pravým kliknutím na item je zobrazena kontextová nabídka s akcemi: upravit, smazat a nový (při kliku na rodičovský item). Kliknutím na item se v hlavním panelu zobrazí výsledek testu.

- Panel akcí:
    - Tlačítko „Obnovit“: Obnoví strom testů. Vyresetuje styl itemů do původního stavu.
    - Tlačítko „Spustit“: Spustí testování výběru testů.
    - Progressbar: Ukazuje aktuální průběh testů.
    - Tlačítko „Zastavit“: Pokusí se zastavit probíhající testování.
    - Tlačítko „Nastavení“: Zobrazí okno s nastavením aplikace.
- Hlavní panel: Zobrazuje výsledek testu. Dobu trvání testu. Dobu trváni requestu. Zobrazuje tělo response, úspěšné testy a neúspěšné testy.

5.3 Nastavení

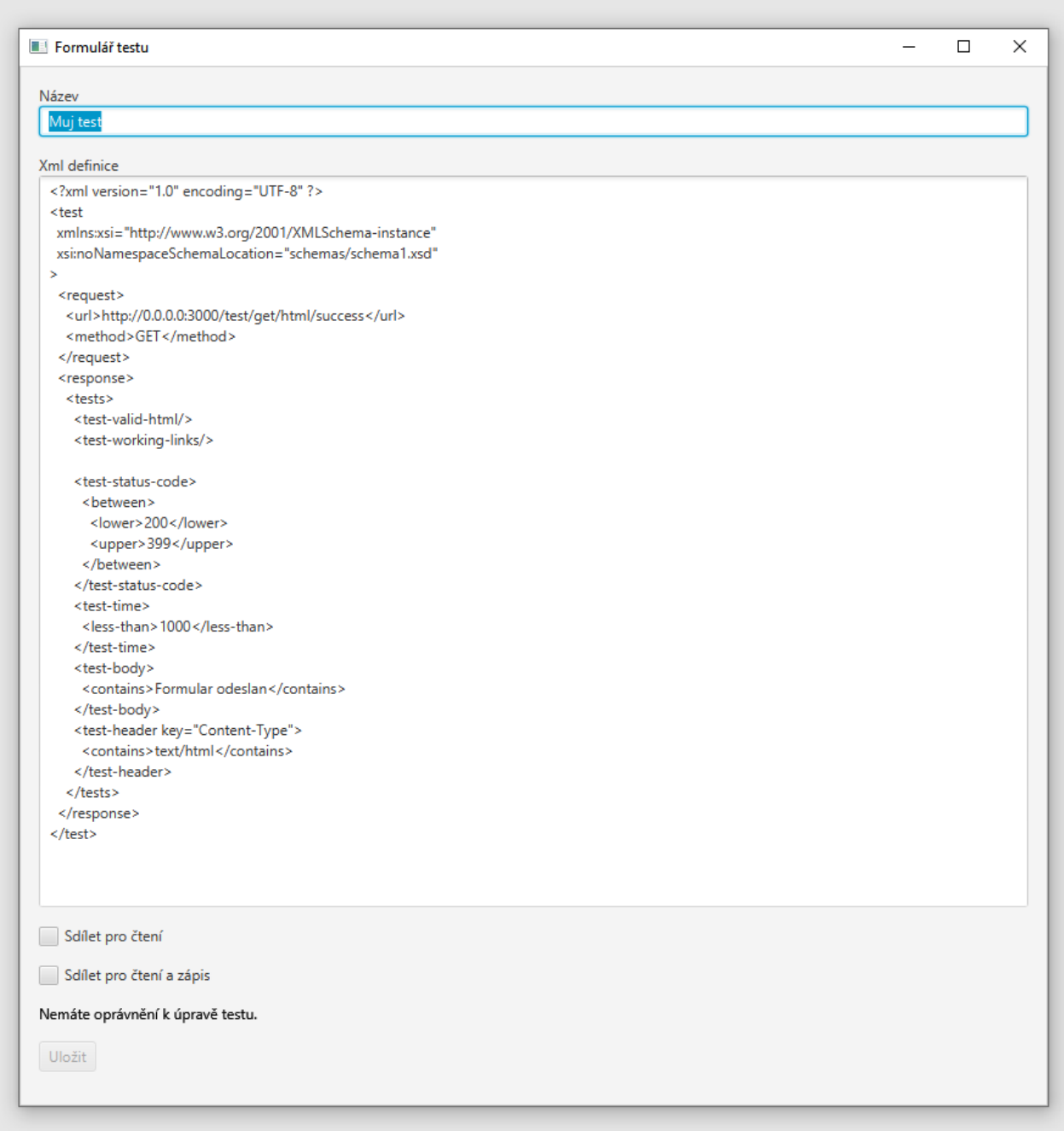
V oknu nastavení je možné zapnout/vypnout ukládání odpovědí a nastavit cestu.



Obrázek 4 okno nastavení

5.4 Formulář testu

Umožňuje vytvořit/upravit test. Pokud uživatel není oprávněn upravit sdílený test, je mu zobrazena hláška a tlačítko je znefunkčněno.



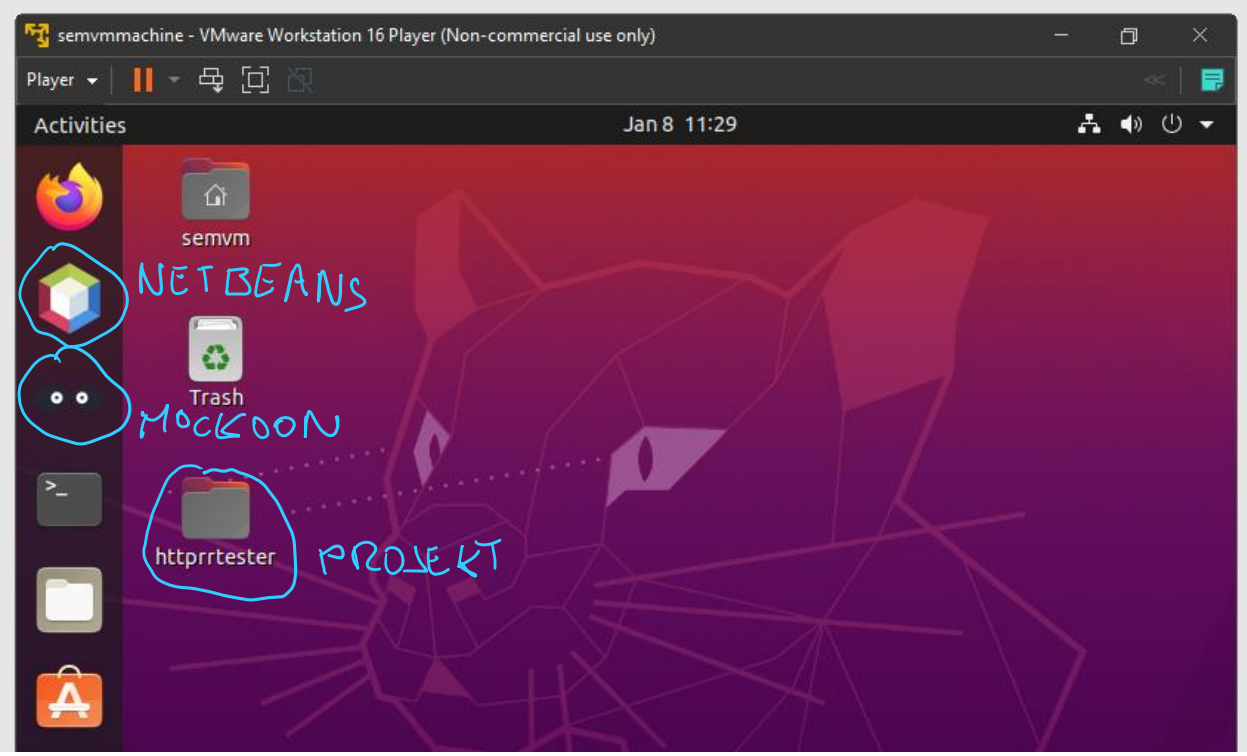
Obrázek 5 okno s formulářem testu

6 Návod na zprovoznění aplikace

6.1 Aplikace

Pro hladké spuštění aplikace jsem připravil virtuální stroj s Ubuntu v programu VMware Workstation 16 Player. Virtuální stroj je k dispozici ke stažení na tomto odkazu https://drive.google.com/drive/folders/1zpVZ9vBgL4K3gdIe-kxWHfK38W15yEIX?usp=sharing. Je nainstalována java-jre, java-jdk v11, Netbeans a Mockoon (více o aplikaci Mockoon v podsekci „Testování“ níže). Předpřipravené existující testy v GUI také používají mock endpointy aplikace Mockoon, takže pro jejich úspěšné provedení je také potřeba mít zapnuto. Projekt je na ploše ve složce „httprrtester“. Stačí jen zapnout Netbeans, otevřít a spustit projekt na ploše (je možné, že postačí jen otevřít Netbeans a projekt tam již bude). Vstupní bod aplikace je přihlášení k databázi – více níže v podsekci „Databáze“.

- Přihlašovací jméno a heslo do Ubuntu: semvm, semvm



Obrázek 6 virtuální stroj s Ubuntu

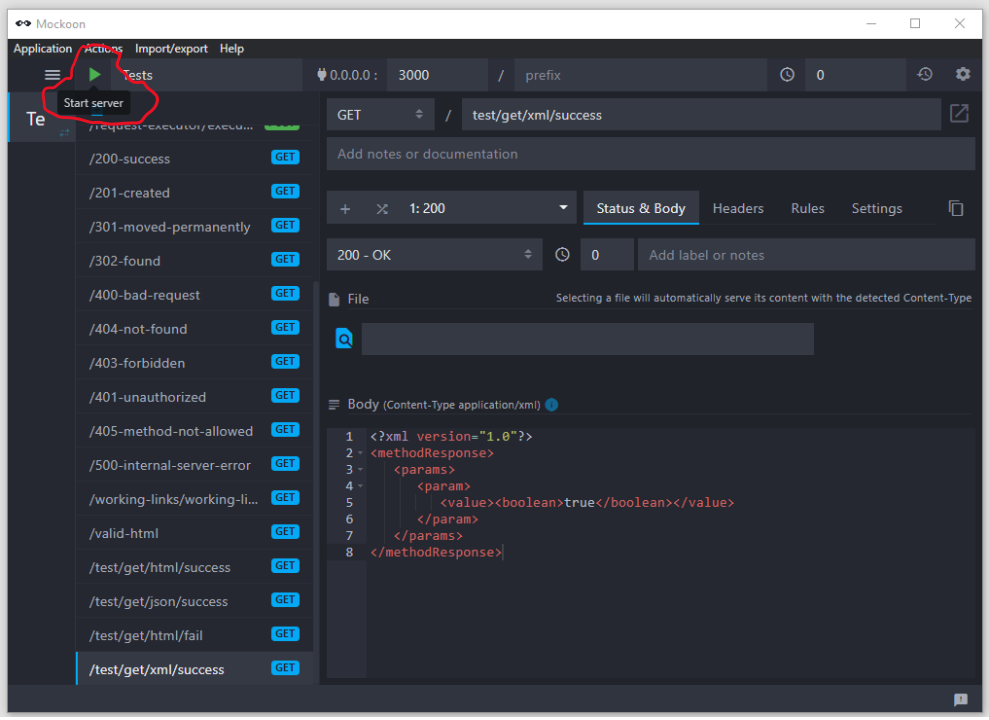
6.2 Databáze

Postgres databáze je pro jednoduchost zprovoznění aplikace připravena na cloudu. Připojovací údaje ke cloud databázi jsou předvyplněny v přihlašovacím oknu. Případně je možné připojit se k jiné databázi. SQL DDL příkazy pro vytvoření struktury jsou přiloženy ve složce projektu v souboru „dbddl.sql“ a příkazy pro vytvoření autorizace „dbauthorization.sql“. V již připravené cloud databázi jsou vytvoření uživatelé pro přihlášení:

- Přihlašovací jméno, heslo: user1, user1
- Přihlašovací jméno, heslo: testuser, testuser

6.3 Testování

Některé testy v aplikaci potřebují pro svůj chod mockované endpointy. Ty jsem vytvářel v aplikaci Mockoon. Tu jsem předinstaloval ve virtuálním stroji a umístil na plochu. Po zapnutí aplikace stačí spustit kolekci „Tests“ na portu 3000 viz obrázek níže.



Obrázek 7 mockoon spuštení mock endpointů