Tester

Webová aplikace pro zkoušení sebe sama s optimalizací pro mobilní zařízení

Filip Kastl

Úvod do problematiky a analýza

Své zápisky v hodinách si dělám typicky digitálně formou otázka-odpověď (viz př. 1). Tuto metodu jsem vyvinul již v první ročníku a velice jsem si ji - společně s některými svými spolužáky - oblíbil. Záměr mého maturitního projektu byl vytvořit grafickou aplikaci, které by byly jako vstup zadány zápisky právě ve formátu otázka-odpověď a která by byla schopna uživatele z těchto otázek vyzkoušet.

```
Kdy se odehrála Bartolomějská noc?
    1572
Co se odehrávalo ve Francii v době Bartolomějské noci?
    Náboženské války
Kdo vládl na konci náboženských válek?
    Jindřich Navarský IV.
Kdy vládl Jindřich Navarský?
    1589 - 1610
Kdo ukončil náboženské války?
    Jindřich Navarský
- př. 1
```

Takovýto program jsem se snažil implementovat již dříve a ač byly vzniklé verze používané a pro svůj účel dostačující, žádná z nich nesplňovala zcela představy ani moje, ani dalších uživatelů. Jejich hlavním nedostatkem bylo to, že byly vázané buď na platformu Python nebo Java a tudíž je nebylo možné spustit na mobilních zařízeních. Dále, každou novou verzi jsem musel ručně rozesílat mezi uživatele programu. Program také neměl žádnou zabudovanou funkci na sdílení zápisků.

Všechny tyto nedostatky mají negativní vliv na flexibilitu aplikace. Způsobují, že se uživatel nemůže nechat vyzkoušet kdekoliv a kdykoliv je mu to vhodné. Musí mít u sebe svůj notebook, můj program a i textový soubor se svými zápisky. To kupříkladu na cestě tramvají nemusí být možné. Zmíněné nedostatky má můj maturitní projekt vyřešit a přidat nové funkce, které dřívější 'testery' nenabízely.

K problému jsem přistoupil následovně: Rozhodl jsem se Tester vypracovat jako responsivní webovou aplikaci. Díky tomu ho lze používat nejen na osobním počítači, ale i na mobilním zařízení a to bez újmy na 'user-experience' (ovladatelnosti). Uživatel u sebe také nepotřebuje mít binární soubor programu, chce-li se nechat vyzkoušet. Stačí mu webový prohlížeč a připojení k internetu - obojí dnes široce rozšířené. Není třeba rozesílat nové verze programu. Stačí aktualizovat verzi běžící na webovém serveru. Uživatel ani nemusí rozesílat své zápisky jako textové soubory. Místo toho je nahraje do aplikace a s ostatními sdílí pouze webový odkaz.

Můj maturitní projekt má být užitečný hlavně dosavadním uživatelům mých 'testerů'. S trochou štěstí by si však mohl najít uživatele nové právě díky tomu, že spustit ho vyžaduje pouhé kliknutí na webový odkaz.

Instalace

Aplikace je uzpůsobená tak, aby běžela na OS Linux. Pro její spuštění je třeba mít nainstalovaný Yarn package manager se všemi jeho dependencies včetně Node.js a Python3 interpret s balíčkem 'virtualenv'.

Instalace dependencies

Python dependencies

Vejděte do kořenového adresáře projektu a spusťte následující příkazy. Ty v adresáři vytvoří 'virtual enviroment', aktivují ho a nainstalují do něj všechny Python balíčky, které aplikace vyžaduje.

```
python3 -m pip install virtualenv
python3 -m venv .env &&
source .env/bin/activate &&
pip install -r requirements.txt
```

Node.js dependencies

Spusťte následující příkaz, kterým nainstalujete všechny Node.js balíčky, které aplikace vyžaduje.

```
yarn install
```

Vygenerování souborů serveru

Spusťte následující příkazy. Zkompilují JavaScript a nechají Django vytvořit databázi serveru.

```
yarn build
python3 ./tester/manage.py makemigrations backend
python3 ./tester/manage.py migrate
```

Spuštění serveru

Následující příkaz spustí Django server pro localhost. K webové aplikaci následně půjde přistoupit přes adresu localhost:8000.

```
python3 ./tester/manage.py runserver 0.0.0.0:8000
```

Aby byla aplikace přístupná i z jiných zařízení než ze hostujícího počítače, je třeba pomocí wsgi propojit lokální Django server s dedikovaným serverem jako je např. Apache nebo nginx.

Před tím, než bude aplikace používána tímto způsobem, by měl být pro Django vygenerován nový secret key.

Implementace

Knihovny

Backend webové aplikace zprostředkovává framework Django s pluginem Django REST framework, který usnadňuje implementaci komunikace backendu a frontendu. Programovacím jazykem pro backend je Python3.

Frontend je vystavěný pomocí frameworků React a Material UI. Dále používám Babel na kompilování JavaScriptového kódu a Webpack pro práci s JavaScriptovými moduly. Frontend je naprogramován pomocí JavaScriptu.

Standardní a vybírací sady

Kromě standardních sad otázek popsaných v úvodu tohoto dokumentu Tester nabízí i typ sady, ve které uživatel vybírá z nabídky více odpovědí tu správnou.

Zadávání zápisků

Sadu otázek vytvoří uživatel tak, že aplikaci zadá svoje zápisky. K tomu slouží komponenta textového editoru, do kterého může buď uživatel text ručně vepsat nebo zkopírovat ze souboru, kde má své zápisky uložené. Text je následně odeslán na server, který ho zparsuje a uloží do databáze jako sadu otázek.

Formát zápisků

Nejprve je třeba napsat na jeden řádek otázku. Poté lze libovolný počet řádku odsadit tabulátorem a na ně napsat odpověď na otázku. V případě vybírací sady, kde otázky mívají více odpovědí, však bude každý řádek považován za jinou odpověď. Z nich tu první si aplikace zapamatuje jako tu správnou. Odpověď (nebo odpovědi) na jednu otázku končí tam, kde začíná již neodsazený řádek s novou otázkou. Typický zápis pro standardní sadu otázek lze vidět na př. 1 v úvodu tohoto dokumentu. Ukázka zápisu pro vybírací sadu otázek je pak zde na př. 2.

```
Kdo napsal Linux kernel?
    Linus Torvalds
    Ken Thompson
    Richard Stallman
    Dennis Ritchie
    Andrew S. Tanenbaum
Kolik je 2 + 2?
    4
    13
    5
- př. 2
```

Uživatelé

Systém uživatelských účtů není implementován. Jeho funkci zastávají odkazy. Standardně by se každý uživatel musel před použitím aplikace přihlásit a aplikace by musela sledovat, který uživatel má přístup k jaké sadě otázek. Místo toho je po vytvoření sady otázek vygenerován unikátní kód. Pomocí tohoto kódu pak tvůrce sady k ní přistupuje. Chce-li, může sám kód sdílet s jinými uživateli a tak jim k sadě také umožnit přístup.

Bezpečnost

Logika zkoušení probíhá výhradně na frontendu. Backend pouze poskytne sadu otázek a po skončení testu zapíše jeho výsledek. Toho je možné zneužít a aplikaci odeslat falešný výsledek. To však není vada projektu. Projekt je určen pro zkoušení sama sebe, ne zkoušení např. žáků učitelem. Zasíláním falešných výsledků klame uživatel pouze sám sebe. Přístup, který jsem zvolil, není tolik závislý na stabilitě připojení k serveru.

Standardní sada

Po tom, co je zobrazena otázka, měl by si uživatel rozmyslet její odpověď a následně otázku rozkliknout. Stránka pak odhalí odpověď a uživatel dle svého svědomí pomocí tlačítek "vím" a "nevím" aplikaci sdělí, zda správnou odpověď na otázku opravdu znal. Podle toho se buď sníží nebo zvýší číslo, kolikrát se ještě otázka v testu vyskytne a uživateli je zobrazena další otázka. Jakmile toto číslo klesne u všech otázek na nulu, test končí.

Tímto způsobem jsou brzy eliminovány otázky jejichž odpověď uživatel pohotově zná. Otázky, se kterými má uživatel problém je naopak nucen zodpovídat vícekrát a tak si je opakovat a snad do budoucna zapamatovat.

Vybírací sada

Při testech z vybíracích sad otázek je správnou odpověď třeba odkrýt jiným způsobem. Společně s otázkou se uživateli zobrazí i náhodně promíchaná tlačítka pro každou možnou odpověď. Úkolem uživatele je stisknout to, které reprezentuje tu správnou. Po tom, co toto učiní, je tlačítko správné odpovědi zvýrazněno a jeho stisknutím se uživatel posune na další otázku.

Sledování výsledků

Uživateli je po dokončení testu zobrazen jeho výsledek, tedy poměr správně zodpovězených a celkově zodpovězených otázek. Dále aplikace zobrazí i průměrný výsledek všech testů provedených na stejné sadě. To je možné, jelikož je výsledek každého testu zaznamenáván. Aplikace by případně později mohla být rozšířena o další funkce, které by i jinými způsoby zpracovávaly tyto výsledky.