Maturitní práce

Informatika

Gymnázium Jana Keplera

Elektronická burza učebnic

Antonín Drdácký

Vedoucí práce: Karel Jílek

Anotace:

Webová aplikace umožňující prodej a nákup učebnic. Po dalších úpravách aplikovatelná na školu gymnázium Jana Keplera.

Abstract:

Web application for sale and purchase of textbooks. After further adjustments applicable to Johannes Kepler Grammar School.

Prohlašuji, že jsem jediným autorem této maturitní práce a všechny zdroje jsou v této práci uvedené.

Tímto dle zákona 121/2000 Sb. ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium Jana Keplera, Praha 6, Parléřova 2 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

Úvod

Cílem projektu je vytvoření elektronické burzy učebnic. Tato burza by fungovala po celý rok na serverech školy a mohla by tak nahradit burzu učebnic, která tradičně probíhá kolem dvacátého září, čili jednou ročně, nebo jí doplnit o celoroční provoz. Význam aplikace ve výměně je domluvit prodej knihy, není jím samotná transakce, ať už knih, či peněz.

Zadání

Vytvořit elektronickou burzu učebnic, na které by bylo možné prodávat a nakupovat knihy. Aplikace by měla obsahovat přihlašovací systém. Pro nepřihlášené uživatele by mělo být možné procházet knihy. Přihlášení navíc mají moct knihy kupovat od ostatních uživatelů a také jim je prodávat. Vytvořit odpovídající databáze, tedy databázi uživatelů, nabídek a knih.

Analýza

GJK zatím nemá žádný takový elektronický systém, který by pracoval s učebnicemi. Již má webové stránky, a tak by se vzhled aplikace měl řídit stránkami, tedy www.gjk.cz. Projekt je tak staven od píky, jen sdílí některé barvy a vzhledové funkce se stránkami školy. Ruce má proto programátor zcela volné.

Návrh

Návrh aplikace

Aplikace by měla splňovat následující teze. Uživatelé mohou systémem procházet, i když jsou nepřihlášeni. Mají přístup ke všem nabídkám, mohou si je prohlédnout. Po přihlášení uživatel může na nabídky přihodit a následně proběhne transakce. Také mohou knihu prodat výběrem knihy ze seznamu, určením ceny a stavu. Pokud se kniha v seznamu nenachází, přihlášený uživatel jí do něj může přidat. Uživatel též může prohlížet

již provedené transakce v historii transakcí. Aplikace by měla být do budoucna rozšiřitelná.

Použité technologie

Aplikace byla naprogramována v Pythonu3, pomocí knihovny Django. Python je vyšší programovací jazyk přívětivý pro programátora. Knihovna Django umožňuje vývoj a správu aplikace. Pro ukládání byla použita PostgreSQL databáze. Oproti SQLlite, základní databází používanou Djangem, je mnohokrát rychlejší a její instalace je snadná.

Návrh databáze

Databáze se skládá ze tří (čtyř) modelů, které odpovídají tabulkám. Je tu User, který dědí z Django třídy User, obohacený jen o třídu (myšleno ročník), do které žák chodí. Pro knihy používám model Book, z níž jsou řádky ISBN a subject zatím nevyužity, jinak obsahuje obrázek, jméno autora a název knihy. Ve vztahu ManyToOne se modul Offer vztahuje k modulu Book.

Offer dědí z AbstractOffer, do kterého jsou vyčleněny některé sloupce, které by sdílel s Demand, který jsem se rozhodl neimplementovat. Offer obsahuje všechny nabídky, nabídka je vždy na prodej. Systém by se dále dal rozšířit o modul Demand dědící z AbstractOffer, který by obsahoval poptávku po knihách.

Dokumentace

Postup instalace

Požadavky pro instalaci

- Python3 (s pipem)
- PostgreSQL
- instalace je dělána pro Windows, na ostatních OS nemusí fungovat

Instalace

1. Ujistěte se, že složka obsahující projekt se jmenuje BurzaGJK.
2. Připojeni k internetu instalujte knihovny pomocí ‚pip install -r requirements.txt‘ ve složce BurzaGJK.
3. Ve složce BurzaGJK vytvořte ‚local_settings.py‘ podle vzoru ‚local_settings_default.py‘
4. Vyplňte DATABASE a SECRETKEY[1] v ‚local_settings_default.py‘.
5. Migrujte databázi pomocí příkazu ‚python manage.py migrate‘
6. Pro správu můžete vytvořit superusera ‚python manage.py createsuperuser‘
7. Server spustíte pomocí příkazu ‚python manage.py runserver‘ ve složce BurzaGJK.
8. Pro test můžete přidávat dummy_data pomocí superusera po spuštění serveru na stránce ‚localhost:8000/admin/‘. Usery tam kvůli heslu nedělejte (je potřeba hash), namísto toho je vytvářejte pomocí příkazu ‚creatueuser‘ v shellu. (Programátorská dokumentace).

1 SECRETKEY lze vygenerovat např. zde https://www.miniwebtool.com/django-secret-key-generator/

Projekt na webu

Projekt se nachází na GitHubu. Udělal jsem verzi k tomuto dokumentu https://github.com/Brambor/BurzaGJK/releases/tag/1.0, neboli ten samý kód, který byl odevzdán na CD.

Přihlášení do aplikace

Do aplikace se dostanete přes localhost:8000, jak už vám řeklo Django v příkazovém řádku. Nového uživatele vytvoříte pomocí příkazu ‚createuser' (programová dokumentace). Pozor, aplikace (přihlašování a následné používání) není určeno pro superusery, i když se na ně lze přihlásit.

Horní lišta

V horním menu je pro nepřihlášené ‚procházet knihy' či ‚nabídky' a tlačítko pro ‚přihlásit se'. Pro přihlášené přibyde ‚prodáváno', ‚přihozeno', ‚k vyřízení', ‚historie' a ‚odhlásit se'.

Všechno to jsou stránky generované viewy, které jsou si podobné tím, že aplikují různé filtry a používají trochu jiné templaty. Kromě těchto drobných odlišností ‚procházet knihy', který volá ‚cluster_list' je zajímavý tím, že, na rozdíl od klasických nabídek view nabídky shlukne a podá koncentrovanou informaci o nabídkách týkající se knihy. Koncentrovanou informací je rozmezí cen a množství v kusech.

Všechny přístupné stránky lze rozdělit do kategorií listy a detaily.

List

Proklikem libovolného z horních tlačítek, kromě ‚přihlásit/odhlásit se' se dostanete na některý z listů. Listy vyberou relevantní nabídky, přes které v templatě iterují. Proklikem libovolné nabídky se dostanete na její detail.

Detail

V detailu se zobrazují podrobné informace. Pokud je možná akce, tak nějaké tlačítko, které akci způsobí. Tlačítko se vždy váže k formě. Akce jsou následující:

- Přes ‚procházet nabídky/knihy' se proklikáte na detail. Zde se nachází tlačítko koupit. Po kliknutí se změní na intuitivní text ‚Tuto knihu kupujete', také se vám objeví v listu ‚přihozeno' a prodejci se objevíte vlistu ‚k vyřízení'.
- V prodáváno můžete přidat knihu proklikem odkazu ‚Přidejte jí!'. Zde se nachází formulář pro přidání knihy. Pokud knihu přidáte, objeví se v seznamu knih.
- V prodáváno dále můžete přidat nabídku vyplněním formuláře. Jedině pole ‚cena' a ‚kniha' jsou povinná. Po odeslání formuláře se kniha objeví v listu prodáváno, umístěného pod formulářem. Formulář:
  - Cena je přirozené číslo.
  - Zaškrtnutím smlouvatelné se za cenou objeví string „(smlouvatelné)".
  - Vyberete knihu ze seznamu knih.
  - Popíšete stav („počmáraná", „jako nová"...).
  - Do ‚K odebrání od' můžete vyplnit datum, od kdy jste ochotni knihu prodat. Takže už v pololetí můžete naházet své knihy na web a vyplnit, že jste ochotni prodat je od srpna.
- A nakonec můžete v ‚prodáváno' prokliknout prodávané nabídky, kde je v detailu lze ‚Stáhnout z prodeje'. To nabídku smaže.
- V ‚k vyřízení' se nachází listy ‚prodat' a ‚koupit', ze kterých se proklikem na knihu (pokud tam nějaká je) dostanete na detail prodeje a koupě.
  - V detailu prodeje je dvoufázový systém. V detailu koupě je jen druhý krok (potvrdíte, že transakce proběhla).
    1. Knihu prodáte někomu z listu ‚Knihu chtějí koupit'. Kliknutím na prodat se nabídka schová z burzy a nakupujícímu se přidá do listu ‚k vyřízení'.
    2. Jakmile proběhne transakce, systému to sdělíte zmáčknutím tlačítka ‚transakce proběhla'. To vás odkáže zpět na ‚k vyřízení' a nabídka se přesune do vaší historie.
- V ‚přihozeno' ani v ‚historie' žádné akce nejsou.

Programátorská dokumentace

Přidávání Uživatelů

1. Ve složce BurzaGJK spusťte shell. To uděláte tak, že do command liny napíšete příkaz ‚python manage.py shell‘. (Ve složce BurzaGJK!)
2. Pro importování modelu User napište ‚from shop.models import User‘
3. Pro vytvoření nového uživatele napište ‚User.objects.create_user('john', 'lennon@beatles.cz', 'johnpassword')‘
4. Po stisknutí enteru se vytvoří uživatel „john“. Podle vzoru vytvářejte další uživatele.

Implementace

Aplikace běží na Django[2] serveru, který je napojen na PostgreSQL[3] databázi, ke které jste nastavili připojení v BurzaGJK/local_settings.py. Django je knihovna Pythonu[4] Možná jste zvolili jinou databázi. Upozorňuji, že je pak zbytečná Python knihovna psycopg2, kterou jste nainstalovali spuštěním příkazu ‚pip install -r requirements.txt‘ během instalace.

Urls[5]

BurzaGJK/urls.py je url dispatcherem. Stará se o zavolání příslušného viewu při http requestu.

Views[6]

Viewy se nacházejí v BurzaGJK/shop/views.py a jsou to pythonovské funkce, které se starají o generování html, které je vráceno na http dotaz. Dělají tak pomocí ‚return HttpResponse(template.render(context, request))‘

2 https://docs.djangoproject.com/en/2.0/
3 https://www.postgresql.org/docs/
4 https://www.python.org/doc/
5 https://docs.djangoproject.com/en/2.0/topics/http/urls/
6 https://docs.djangoproject.com/en/2.0/topics/http/views/

na kocni každého viewu, kde ‚context' jsou data, která funkce předává do templaty.

Jsou tu function-based viewy, které se dnes již nehodí pro Django aplikace, jelikož class-based viewy zredukují množství duplikátního kódu. Proto by aplikace měla viewy nahradit class-based viewy před uvedením do produkce, a to kvůli spravovatelnosti aplikace.

Pro viewy v této aplikaci obecně platí, že view obsahující „_list" se stará o generování listu a view obsahující „_detail" se stará o generování detailu, které jsou vysvětleny v ‚Dokumentace', konkrétně ‚List' a ‚Detail'.

Na začátku každého viewu, který potřebuje omezit uživatele, nebo zjistit, zda je uživatel přihlášen a který je přihlášen se pomocí ‚if-else' statementu vyřeší dotaz na základě sessions, které jsou uloženy v requestu, takže se view ptá po ‚request.user.is_authenticated'.

Před produkcí by se měly tyto ‚if-else' statementy nahradit za dekorátory[7]. Opět kvůli redukci duplikátního kódu. Tedy, pokud by se to nevyřešilo už class-based viewy, které jsou přednější.

Dále už se viewy liší jen tím, jak zpracují data z databáze, ke které se dostávají pomocí modelů a data filtrují většinou funkcemi ‚filter' a ‚exclude'. Také se liší tím, jakou templatu použijí. View může přesměrovat na jiný view pomocí redirect, čímž se view ptá zpět na urls.

Vyjímkou filtrování dat je view transact_list, u kterého bylo potřeba použít ‚annotate'. Řádek 231.

Modely[8]

Modely se nacházejí v BurzaGJK/shop/models.py. Umožňují pythonu přístup k databázi, jak na zápis tak čtení dat. Ve složce BurzaGJK/shop/migrations/ jsou jednotlivé transformace mezi verzemi databáze. Při úpravě modelů je potřeba zavolat příkaz ‚python manage.py makemigrations', v comand line, pro generování nových migrací. Pro aplikování migrací je potřeba použít ‚python manage.py migrate', což aktualizuje sloupce databáze podle migrací a momentálního stavu databáze.

7 https://docs.djangoproject.com/en/2.0/topics/auth/default/#the-login-required-decorator
8 https://docs.djangoproject.com/en/2.0/topics/db/models/

Templates[9]

Templaty se nacházejí v BurzaGJK/templates/. Jde o zčásti statické html soubory s Django syntaxí umožňující, aby části souboru byli dynamické. Například pro procházení listů pythonovským ‚for‘ nebo užitím ‚if-else‘ statementu.

Závěr

Projekt splňuje zadání, je funkční, nicméně před nasazením do produkce je potřeba ho dodělat, napříkald co se bezpečnosti týče, a přizpůsobit pro snadnou správu.

Například by bylo možné značně zredukovat kód ve views. Kupříkladu validace uživatelů se neustále opakuje. V aplikaci, která by se dostala do produkce, by navíc mělo být možné přihlásit se přes google autentifikaci užitím školních mailů.

Položek k úpravě je mnoho a plánuji je všechny implementovat před uvedením do produkce. Především nastavit DEBUG = False, a následně nastavit ALLOWED_HOSTS, ujistit se, že Django se stále dostane ke static souborům atd.

9 https://docs.djangoproject.com/en/2.0/topics/templates/