V Teplicích dne 24.3.2022

Informace k odevzdávanému projektu

Rich Insurance

Na základě zadání nepovinného projektu k závěrečné zkoušce a následné prezentaci při hledání zaměstnání jsem zpracoval projekt “informačního systému pro pojišťovny.

Rich Insurance - main page

Přestože jsem se snažil, aby projekt vypadal reálně, některé věci jsem oproti běžně používané praxi v tomto odvětví značně zjednodušil. Momentálně je z mého pohledu mým největším problémem pro odevzdání pokročilejší a propracovanější verze čas. Na aplikaci jsem začal pracovat relativně brzy po započetí studia na ITN a na některých částech kódu je to vidět. Snažil jsem se aplikaci posouvat funkčně dále s primárním zaměřením na funkčnost vůči mým znalostem. V aplikaci chci i po jejím odevzdání pokračovat a upravit vše dle současných dobrých zvyků. Nejvíce jsem zápolil s pro mne dosti rozdílným kódem ve frameworku Django oproti standardnímu Python kódu a dlouho jsem hledal informace, jak vlastně funguje OOP v Djangu. Veškeré Modely jsem se

tak snažil napsat v rámci OOP, ale dnes už rovněž vím, že do class chci předělat i views, aby došlo ke zpřehlednění a redukci kódu. Toto jsem již bohužel nestihl. Rovněž jsem se potýkal s otázkou umístění logiky, kdy samotná dokumentace k Django fr. říká, že Django samotné je vlastně controler. S ohledem ne minimum výpočtů jsem se rozhodl logiku umístit převážně do views, kde sice vznikají výpočty tzv. “On the fly”, ale s ohledem na rozsah aplikace by zde neměl být negativní dopad na odezvu.

Věřím, že v projektu bude stále plno nedostatků, ale v rámci dalšího učení a vzdělávání se, se je na tomto projektu plánuji postupně odstraňovat a nadále se učit. Prozatím musím říci, že tvorba tohoto projektu byla nesmírně důležitá, jelikož jsem díky ní postupně metodou pokus - omyl došel k tomu, že mám alespoň obecnou představu, jak tvorba web aplikací v tomto frameworku funguje a ačkoliv architektura Django není plně dle MVC, tak nyní vnímám, jak aplikaci dle tohoto vzoru rozdělit.

Použitá databáze SQLite : Ačkoliv jsem v rámci aplikace vše propojil do Postgres databáze a vše bylo funkční, aplikaci odevzdávám s napojením na SQLite. Za toto se musím omluvit, jelikož jsem cca dva týdny před odevzdáním začal řešit Master password do Postgres a po následné snaze o jeho smazání a novou instalaci, s tím, že v Djangu udělám nově makemigrations a migrate do nové DTB jsem se dostal do stavu, kdy PG nemohu ani nainstalovat, jelikož někde v OS zůstala nějaká část, kterou nemohu najít. V praxi by to asi řešil docker, zde to vyřeší reinstalace systému, ale tu jsem z časových důvodů chtěl provést až po odevzdání projektu.

Přístupy

Pro názornost a v rámci testování aplikace jsem vytvořil několik “zákazníků”. Jedná se především o mé domácí mazlíčky, takže se prosím nelekněte jejich specifických dotazů atd.

admin	liborek
krecek	liborek
kekel	liborek
blazena	liborek
denisa	liborek
Dolcegusto	liborek

O aplikaci + k dalšímu dopracování

- Aplikace je rozdělena na dvě podaplikace Admin a Customer. Dle loginu má každá role své vlastní specifické views a tím i přidělené možnosti.
- Při návrhu aplikace jsem postupoval tak, aby při jejím ovládání nebylo třeba použít předpřivenou /admin sekci. Počítám s jejím využitím při nastavování admin/user práv.
- V části Customer počítám s doděláním editace přihlášeného zákazníka. Prozatím jsem stihl pouze v sekci admin. Ze sekce admin u editace bude třeba upravit, aby forms nepožadovalo heslo, jelikož to admin v reálu nebude znát.
- V části admin jsem záměrně vynechal tvorbu nového zákazníka. Pokud by admin chtěl zákazníka z nějakého důvodu založit, může tak vždy udělat za pomoci klasické registrace.
- Systém přidělení produktu je nastaven tak, že si klient zažádá o produkt a poradce může schválit/odmítnout. Tím nemůže dojít k přidělení pojištění klientovi, který ho nechtěl. V rámci reálného provozu by asi bylo dobré, aby pojištění klientovi mohl samovolně založit i poradce bez předchozí žádosti.
- Můj první návrh DTB struktury nereflektoval fakt, že každý zákazník může chtít jiné pojistné limity a poradce mu může přidělit libovolnou slevu. Původní myšlenka byla vytvořit několik “balíčků”, ze kterých si každý musí vybrat. Následně jsem upravil tak, aby systém balíčků zůstal, avšak poradce po schválení pojištění u konkrétní vazby produktu na zákazníka nastavoval měsíční pojistné a případnou slevu.