**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**UNIVERZITA KOMENSKÉHO**

**Návrh**

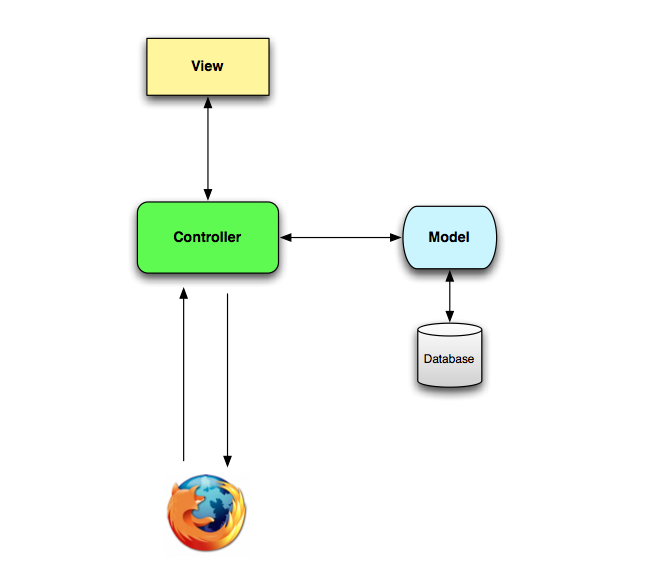
**WEBOVÁ APLIKÁCIA PRE TURISTICKÚ SKUPINU**

|  |
| --- |
| 2015/2016 Kristína Miklošová Juraj Mašlej Juraj Koštál Roman Brojo |

Na naprogramovanie použijeme framework Django. Django podporuje štandartnú MVC architektúru. Návrh tried bude preto ovplyvnený touto architektúrou.

**Model-View-Controller (MVC)**

* **Model** - reprezentuje dátovú štruktúru stránok. Triedy vrstvy Modelu obsahujú zvyčajne metódy pre prístup k dátam, ktoré sú uložené v databáze.
* **View** - sú informácie, ktoré dostáva z controllera a sú prezentované užívateľovi. Nemá priamy prístup k dátam.
* **Controller** - slúži ako prostredník a sprostredkúva premostenie medzi Modelom a View a tiež medzi ostatnými zdrojmi potrebnými.



*Obr.1 diagram architektúry MVC*

**Model**

Prvým krokom pri vytváraní databázovej Django aplikácie je definícia modelov – t.j. štruktúry databázy s dodatočnými metadátami. V našom projekte bude preto jeden model pre každú tabuľku. Napríklad definujeme triedu Zápis, ktorá bude dediť z triedy model. Model bude predstavovať jednu tabuľku v databáze. V tejto triede sa ďalej zadefinujú triedne premenné, ktoré budú predstavovať jeden riadok v tabuľke. Tieto triedne premenné budú odkazovať na jednotlivé triedy typu Field napr. CharField (znakové pole) alebo DateField (datum a čas).

Pomenovaním jednotlivých polí Field chceme definovať tzv. “strojové“ (machine-friendly) meno. Cez ne sa bude v Pythone pristupovať k hodnote poľa a v databáze bude využité pre pomenovanie stĺpca. Pokiaľ by sme chceli, môžeme pomocou prvého argumentu vo Field vytvoriť “ľudský čítateľné“ (human-redable) meno. Ak ho nezadáme, Django si ho odvodí zo strojového mena. Niektoré Field triedy majú povinné parametre, napr. CharField vyžaduje špecifikáciu max\_lenght (maximálna dĺžka reťazca). Django cez Triedu podporuje takisto všetky obvyklé databázové relácie: N (many-to-many), 1:N (many-to-one) a 1:1 (one-to-one).

Čiže premenná Zápis bude obsahovať triedne premenné:

1. id – vygeneruje sa automaticky, toto správanie sa dá zmeniť

2. nazov – typu CharField

3. datum – typu DateField

4. kraj – typu CharField

5. ucastnici – typu OneToManyField, kde ako parameter bude trieda Zapis\_uzivatel

6. autor – typu OneToOneField, kde ako parameter bude trieda Uzivatel

7. vrstvy – typu OneToManyField, kde ako parameter bude trieda Zapis\_vrstva

8. textovy\_popis – typu TextField

9. gps\_poloha - typu PointField

Zvyšné triedy budú definované podobným spôsobom. V Triede je vhodné dodefinovať rôzne metódy, ktoré využijeme pri tvorbe views. Napr. pri pomocnom vypisovaní funkciou print (objekt) nám bude funkcia vracať výraz <Zapis: Zapis object>. Preto je vhodné dodefinovať špeciálnu metódu \_\_str\_\_, ktorá bude vracať vhodnejšiu reprezentáciu objektu. Napr. pre Triedu Zápis, by funckia return mohla vracať: nazov+“ „+ datum.

**Controller**

V prípade Djanga „controller“ je framerworkom ako takým: mechanizmus, ktorý posiela požiadavky určenému view, podľa Django URL konfigurácie. Preto sa triedy Controller-u v projekte nebudú vytvárať.

**Views**

View je “druh” webovej stránky, ktorá má v našej aplikácií určitú úlohu a výsledok svojej činnosti prezentuje prostredníctvom šablóny. V Djangu je každé view zrealizované jednoduchou funkciou v Pythone. V našej aplikácií bude pre prehľadnosť každé view osobitná trieda (dediť bude z View). V tejto triede bude zadefinovaná špeciálna funkcia get, ktorá ako parameter bude dostávať HTTP request a bude vraciať HTTP response. Django obsahuje takisto šablónový systém. Vo views odovzdá kontext, čo je slovník, ktorý mapuje šablónové premenné na objekty Pythonu. Vo views sa takisto môžu objaviť základné prvky Pythonu ako for loop alebo if statement. V našej aplikácií bude každa url adresa so stránkou predstavovať 1 triedu typu View.

**LogIn**

Bude využívať  Django’s authentication system.