Správa o realizácii projektu na predmet: Objektovo orientované programovanie

Názov : VOUT – digitálny hlas Slovenska

Meno : Anna Hollá, FIIT STU

Zámer projektu:

VOUT je softvérová platforma zameraná a vytvorená na zjednodušenie a demokratizáciu volieb pre občanov Slovenskej republiky. Za dôvodom vzniku tohto systému stojí aj neustále sa zvyšujúci trend vo veľmi nízkej volebnej účasti, na ktorý vplývajú mnohé faktory. Medzi tieto patria napríklad: neflexibilita, časová náročnosť, nedôvera voči spracovaniu dát, či mnohé iné.

Systém VOUT dbá najmä na odstránenie týchto problémov pomocou inovácie doposiaľ používaného volebného systému, s prihliadnutím na bezpečnosť, transparentnosť a digitalizáciu dát. Pokryté sú štyri typy volieb - komunálne, parlamentné, prezidentské a voľby do Európskeho parlamentu. V prípade konania jedných zo spomínaných volieb má každý občan, spĺňajúci štandardné podmienky, ktoré sú automaticky overené pri prihlásení, prístup do systému. Môže tak plnohodnotne odovzdať svoj volebný hlas len na základe jednoduchého procesu prihlásenia, pomocou čísla občianskeho preukazu.

VOUT je napojený na systém štatistického úradu, ktorému priebežne v reálnom čase odosiela dôležité štatistické údaje a zabezpečuje kontinuálnosť volebného procesu.

Platforma VOUT teda ponúka úplne nový, otvorený a demokratický prístup k voľbám na Slovensku. Spĺňa a kopíruje požiadavky dnešnej digitálnej doby a uľahčuje tak hlasovanie každému občanovi.

Štruktúra projektu :

Projekt je založený na základnom princípe objektovo orientovaného programovania = interakcia objektov. Triedy jednotlivých objektov sú zmysluplne zoradené do hierarchií. Program obsahuje celkom 5 hierarchií, z ktorých 1 je hierarchia tried Controllerov, ktorá rieši event handlery pre grafické užívateľské rozhranie (GUI). Ostatné hierarchie sú systémové, riešia už len aplikačnú logiku. Na rozhraní medzi GUI a aplikačnou logikou je hierarchia Service.

Tu uvádzam diagram 4 významných hierarchií tried:

Diagram 1

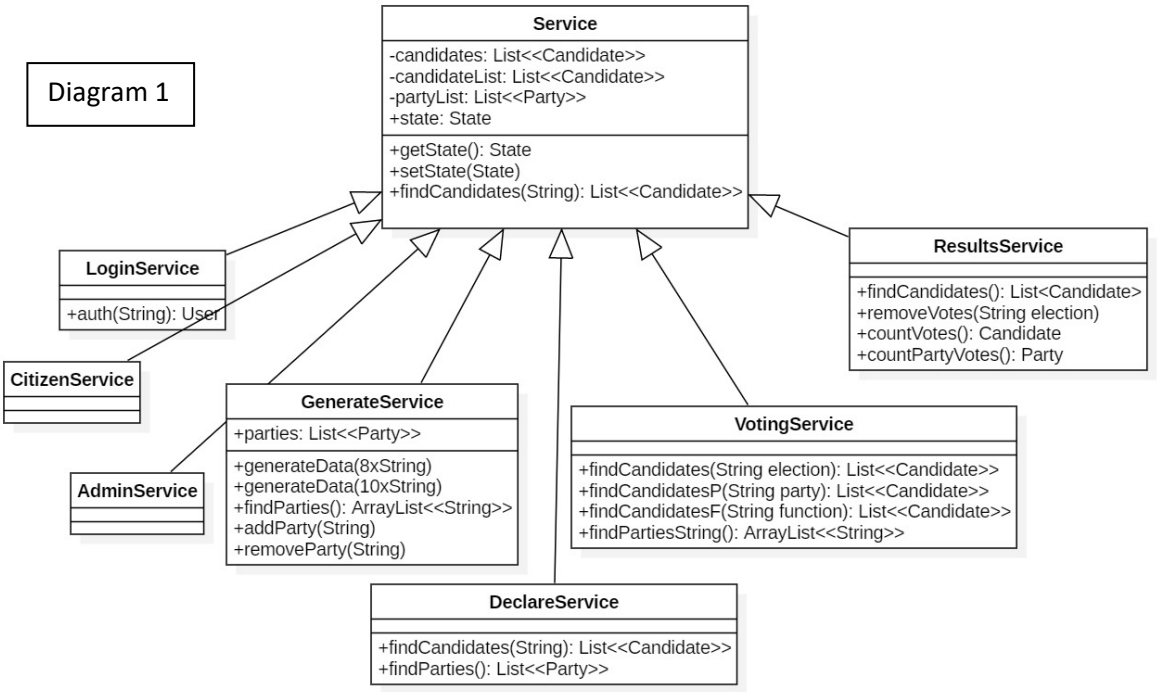

Diagram 2

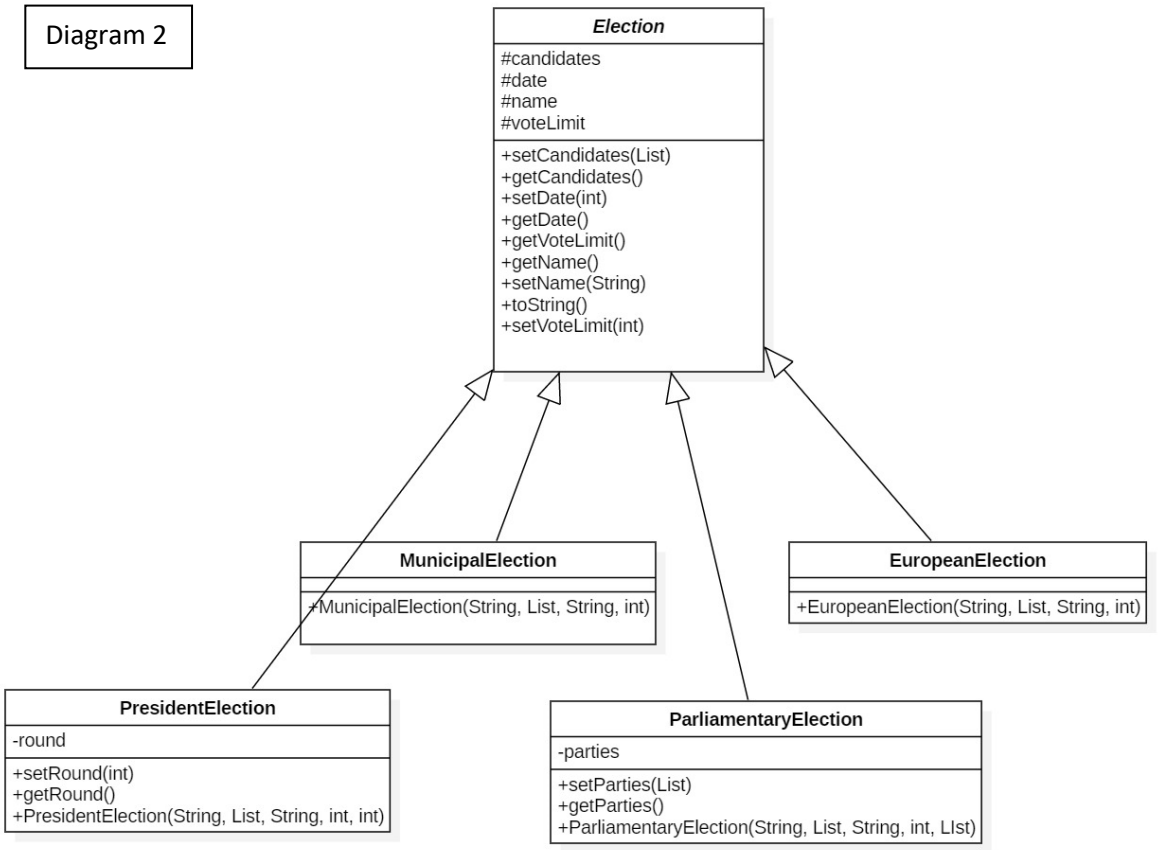

Diagram 3

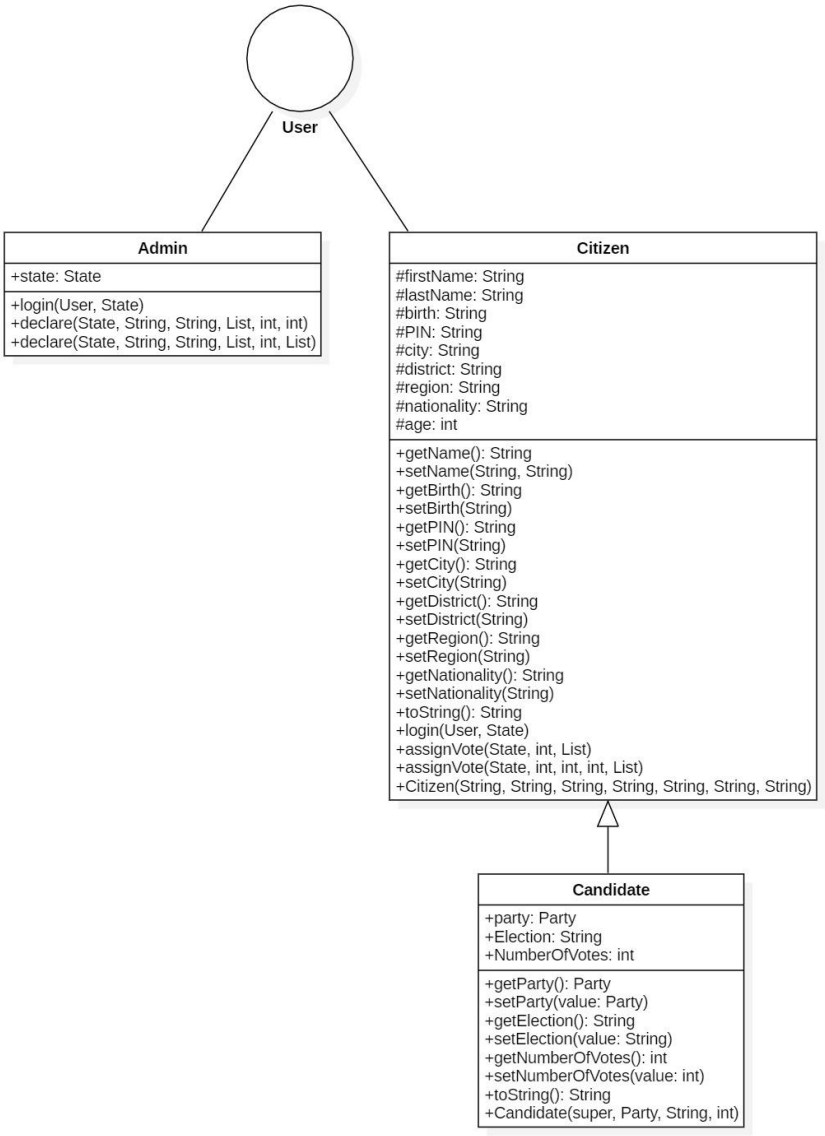

Diagram 4

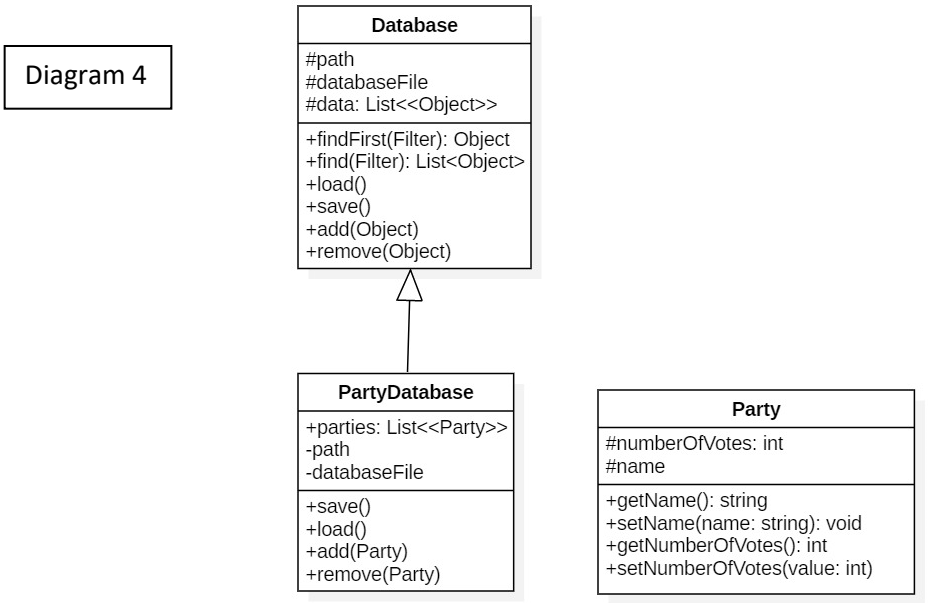

DIAGRAM 1

Diagram znázorňuje hierarchiu tried Service. Od hlavnej triedy Service je oddedených 7 tried. Každá z nich prislúcha jednému z controllerov v GUI package.

Aby sa oddelila aplikačná logika, v controlleroch sa rieši len načítanie vstupu a vypísanie výstupu na jednotlivé komponenty stageov. Akákoľvek funkcionalita sa rieši v príslušnom Service, ktorý si najprv vždy vezme aktuálny stav „State“ programu, ktorý je držaný v objekte State. Následne vykonáva potrebné metódy za pomoci atribútov State.

Táto hierarchia obsahuje ako agregáciu tak aj polymorfizmus.

DIAGRAM 2

Diagram znázorňuje dedenie od abstraktnej triedy Election. Oddedené sú 4 triedy podľa 4 typov volieb. Ich spoločné atribúty ako aj settery a gettery sú v abstraktnej materskej triede. Okrem setterov a getterov obsahuje každá trieda konštruktor, podľa svojich parametrov. Keď za počas behu programu vyhlásia voľby, vytvorí sa objekt konkrétneho typu volieb a uloží sa ako atribút objektu triedy State.

DIAGRAM 3

Diagram 3 znázorňuje interface User, ktorý implementujú 2 triedy: Citizen a Admin, keďže pri prihlásení a overovaní prihlasovacích údajov sa vytvorí práva objekt jednej z týchto tried. Podľa toho aký objekt sa vytvorí, zavolá sa metóda login (pochádza z interface User) v príslušnej triede, a nastaví ako atribút objektu State aktuálne prihláseného používateľa.

Od triedy Citizen je oddedená trieda Candidate, keďže každý kandidát je aj občan a má možnosť voliť. Objekt triedy Candidate má všetky atribúty triedy Citizen a navyše má atribúty: do akej strany patrí (Objekt triedy Party), do ktorých volieb kandiduje a aký má počet hlasov.

Táto hierarchia obsahuje ako agregáciu, tak aj polymorfizmus.

DIAGRAM 4

Diagram znázorňuje dedenie databáz. Od triedy hlavnej databázy „ľudí“ (objekty občania a kandidáti) je oddedená databáza strán (objekty strany).

Obe triedy riešia serializáciu a deserializáciu svojej databázy ako aj pridávanie a mazanie objektov z databázy. Táto hierarchia obsahuje polymorfizmus aj agregáciu.

Splnenie kritérií projektu:

1. HLAVNÉ KRITÉRIÁ
    a. **Dedenie** – program obsahuje niekoľko rôznych hierarchií tried, ktoré využívajú princíp dedenia. Ich vysvetlenie je popísané vyššie pod ich UML diagramami.
    b. **Polymorfizmus** – v hierarchiách sa vyskytuje prekonávanie metód, a to konkrétne:
        1. Prekonanie metód save a load v hierarchii databáz (rozdielna implementácia pri databáze ľudí a strán)
        2. Prekonanie metódy toString v hierarchii entít (rozdielna implementácia pri manipulácií s objektom Citizen, pri ktorom nás vždy zaujíma jeho identita ako prihláseného používateľa, a pri maipulácií s objektom Candidate, pri ktorom nás zaujíma jeho meno a strana, v ktorej kandiduje.
        3. Prekonanie metódy findCandidates v hierarchií Service (rozdielna implementácia pri filtrovaní kandidátov podľa volieb, ktoré sú vyhlásené)
    c. **Preťaženie** metód – v programe sú na viacerých miestach preťažené metódy:
        1. Pri vyhlasovaní volieb v triede administrátor metóda declare = posielajú sa rôzne parametre podľa volieb, ktoré sa vyhlasujú
        2. Pri voľbe občana metóda assignVote = rozdielne parametre podľa toho či sa volia len kandidáti, alebo aj strany.
    d. **Zapuzdrenie** – program v hojnej miere využíva tento princíp, keďže implementácia objektov, ktoré interagujú ostáva skrytá, a dá sa k nim pristupovať pomocou jednotlivých operácií.
    e. **Agregácia** – využíva sa na viacerých miestach v hierarchiách
        1. Trieda Candidate má ako atribút party = objekt triedy Party
        2. Triedy Service majú ako atribút state = objekt triedy State (uchováva aktuálny stav dôležitých častí programu)

2. ĎALŠIE KRITÉRIÁ
    a. **Návrhový vzor Filter** – abstraktná trieda filter implementuje všeobecnú metódu založenú na princípe práce s komponentom HashMap, ktorý porovnáva ľubovoľný kľúč a hodnotu, ktorá je mu zadaná ako parameter. Je to všeobecný algoritmus, ktorý tak vie vyfiltrovať čokoľvek. Od tejto triedy sú oddedené konkrétne filtre, ktoré hovoria o tom, čo sa filtruje (CitizenFilter, CandidateFilter, ...)
    b. **Vlastné výnimky**: v package Service.Exceptions sú 2 vlastné výnimky:
        1. UserNotAuthenticated – vyhodí sa, keď sa chce prihlásiť občan/administrátor so zlými prihlasovacími údajmi
        2. ElectionNotDeclared – vyhodí sa, keď sa chce prihlásiť občan a voliť, pritom ešte voľby neboli vyhlásené
    c. GUI: grafické užívateľské rozhranie, s vlastnými event handlermi v controlleroch. Oddelenie od aplikačnej logiky je dosiahnuté vďaka hierarchií tried Service, ktoré sú prepojené s controllermi.
    d. RTTI: použitím príkazu „instanceof“ sa pri prihlásení rozhoduje, či je osoba, ktorá sa prihlasuje Admin alebo Citizen a na základe toho sa zavolá metóda login v príslušnej triede.
    e. Rozhranie = interface: interface User, ktorý implementujú triedy Citizen a Admin. Obsahuje metódu login, ktorú tieto triedy konkrétne implementujú.
    f. Lambda výrazy: pri filtrovaní sa podľa toho, aký druh porovnávania sa vykonáva, vracia želané poradie objektov (napríklad prvá hodnota je väčšia ako druhá)