Technická dokumentácia pre knižnicu Core

Verzia 16.11.2015

Tabuľka 1. Autori

Autor	Rola
Peter Halaš	Autor

Tabuľka 2. História zmien

Verzia	Dátum	Autor	Popis
1.0	16.11.2015	Peter Halaš	Vytvorenie dokumentu
1.1	24.11.2015	Peter Halaš	Upravenie dokumentu
1.2	29.11.2015	Peter Vrana	Pridanie dokumentácie ku Core.CentralServices
1.3	30.11.2015	Peter Halaš	Upravenie dokumentácie ku Core.CentralServices
1.4	30.3.2016	Peter Vrana	Refaktoring REST služieb v admin portáli a s tým súvisiace zmeny, autorizácia služby pre logovanie
1.5	17.4.2016	Ivan Beňovic	Pridanie dokumentácie ku Core. CentralServices konfiguračnej časti

Obsah

1 Úvod

V dokumente sú opísané zmeny knižnice Core, v rámci tímového projektu DevActs, ktorá je hlavnou knižnicou. Využíva ju viacero modulov architektúry projektu PerConIK. V tejto knižnici bolo potrebné vykonať viacej zmien.

2 Analýza

2.1 Core.Log

Tento modul obsahuje obalenú knižnicu použitú pre logovanie NLog. V súčasnom stave sa loguje do súboru a na konzolu. Na to slúžia oddelené triedy. *ConsoleLogger* a *NLogLogger*. Úlohou je pridať do *NLogLogger* triedy metódu, ktorá bude posielať logy do Administračného portálu cez pripravenú REST službu, aby sa zabránilo zneužitiu tejto služby je potrebné, aby všetky požiadavky boli rozšírené o autentifikačnú hlavičku, ktorá sa kontroluje na strane administračného portálu.

Štruktúra logu je predpísaná podľa Administračného portálu. Trieda *Log* obsahuje tieto rleevantné údaje:

- Typ logu: môže to byť Request, Response, Info, Warn alebo Error
- Čas: kedy sa zavolala metóda triedy *NLogLogger*.
- Správu: obsahuje správu spolu s *errorStackTrace*
- Názov modulu: z ktorého bola zavolaná metóda triedy *NLogLogger*.
- Pole objektov: ktoré môžu byť poslané spolu so správou metódam triedy *NLogLogger*

2.2 Core.CentralServices

Cieľom tejto úpravy je vytvoriť centrálny projekt Core.CentralServices, do ktorého budú umiestnené volania služieb, ktoré poskytuje administračný portál. Cieľom tejto úpravy je zjednotiť volanie služieb administračného portálu na jedno miesto. Takáto úprava má hneď niekoľko výhod. Keďže jednotlivé moduly budú využívať zdrojový kód nachádzajúci sa v Core.CentralServices výrazne sa zníži duplicita kódu a zároveň sa predchádza chybám prameniacim z duplicity kódu. Taktiež sa takouto úpravou zvýši prehľadnosť štruktúry projektu. Táto implementácia bude vyžadovať nasledujúce úpravy:

- Vytvorenie Projektu Core.CentralServices, pridanie potrebných konfiguračných nastavení
- Posielanie logov – volanie služby pre posielanie logov do administračného portálu je potrebné presunúť do Core.CentralServices
- Autentifikácia a autorizácia v Asp.net aplikáciách prostredníctvom volania služieb administračného portálu vyčlenená do Core.CentralServices
- Zistenie konfiguračných hodnôt pre moduly podľa kľúčov

3 Návrh riešenia

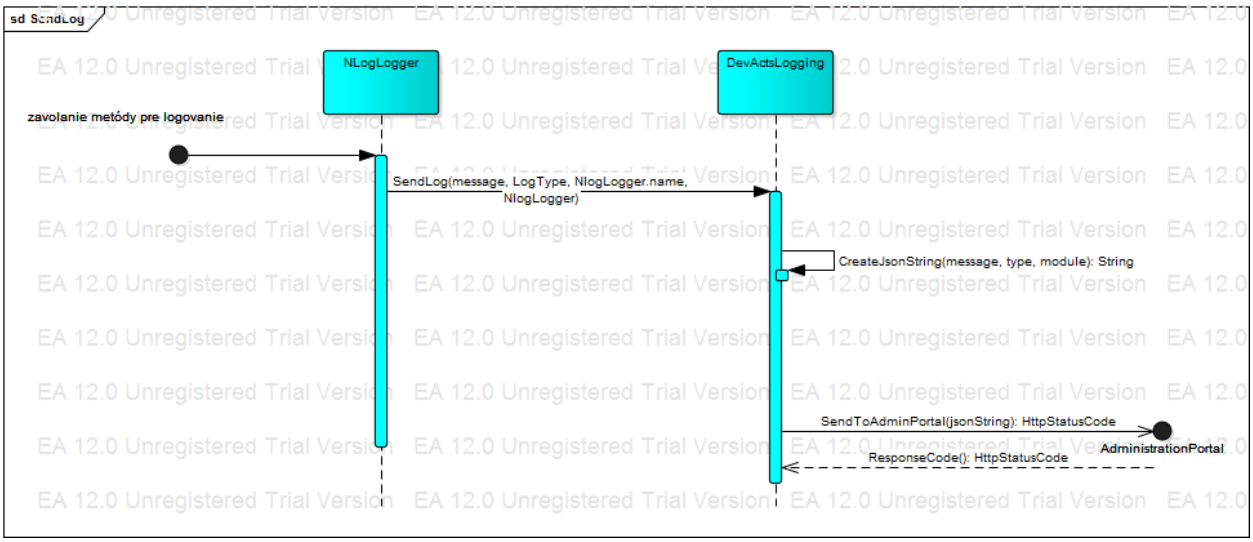
V nasledujúcej časti bude opísané, ako boli navrhnuté jednotlivé úpravy, ktoré boli opísané v časti analýza.

3.1 Návrh posielania logov

Návrh je veľmi jednoduchý, v každom volaní metódy triedy *NLogLogger* sa pridá volanie metódy, ktorá odošle log. Pre každý z 3 typov logov: Info, Warn a Error existujú 2 metódy. Jedna metóda má ako parameter správu, druhá metóda má správu a pole objektov. Preto treba najskôr implementovať metódy ktoré vytvoria log a naplnia relevantné údaje. Následne sa zavolá metóda, ktorá odošle nachystaný log do Administračného portála cez pripravenú REST službu, ktorá akceptuje inštanciu triedy *Log*. Pred odoslaním HTTP požiadavky musí byť táto požiadavka obohatená o autentifikačnú hlavičku tá pozostáva z názvu použitej autentifikačnej schémy, v tomto prípade sa jedná o *Basic*. Názov autentifikačnej schémy je nasledovaný menom a heslom účtu servisného používateľa, jednotlivé položky sú oddelené dvojbodkou. Servisný účet je konfigurovateľný prostredníctvom konfiguračného súboru v Core.CentralServices. Je dôležité, aby sa meno a heslo servisného účtu zhodovalo s menom a heslom použitým v centrálnej správe používateľov v AdminPortáli. Zároveň meno ani heslo servisného používateľa **nesmie obsahovať dvojbodku**, pretože v opačnom prípade dôjde k chybe pri pársovaní autentifikačnej hlavičky v administračnom portáli.

Metódy na vytvorenie a poslanie logu do Administračného portálu sa presunuli do projektu Core.CentralServices, ktorý obsahuje aj iné metódy na volanie služieb (napr. na autorizáciu a autentifikáciu)

Na Obrázok 3.1 je znázornený sekvenčný diagram reprezentujúci základnú postupnosť operácií pro posielaní logov. Volanie REST metódy musí byť vykonávané asynchrónne.



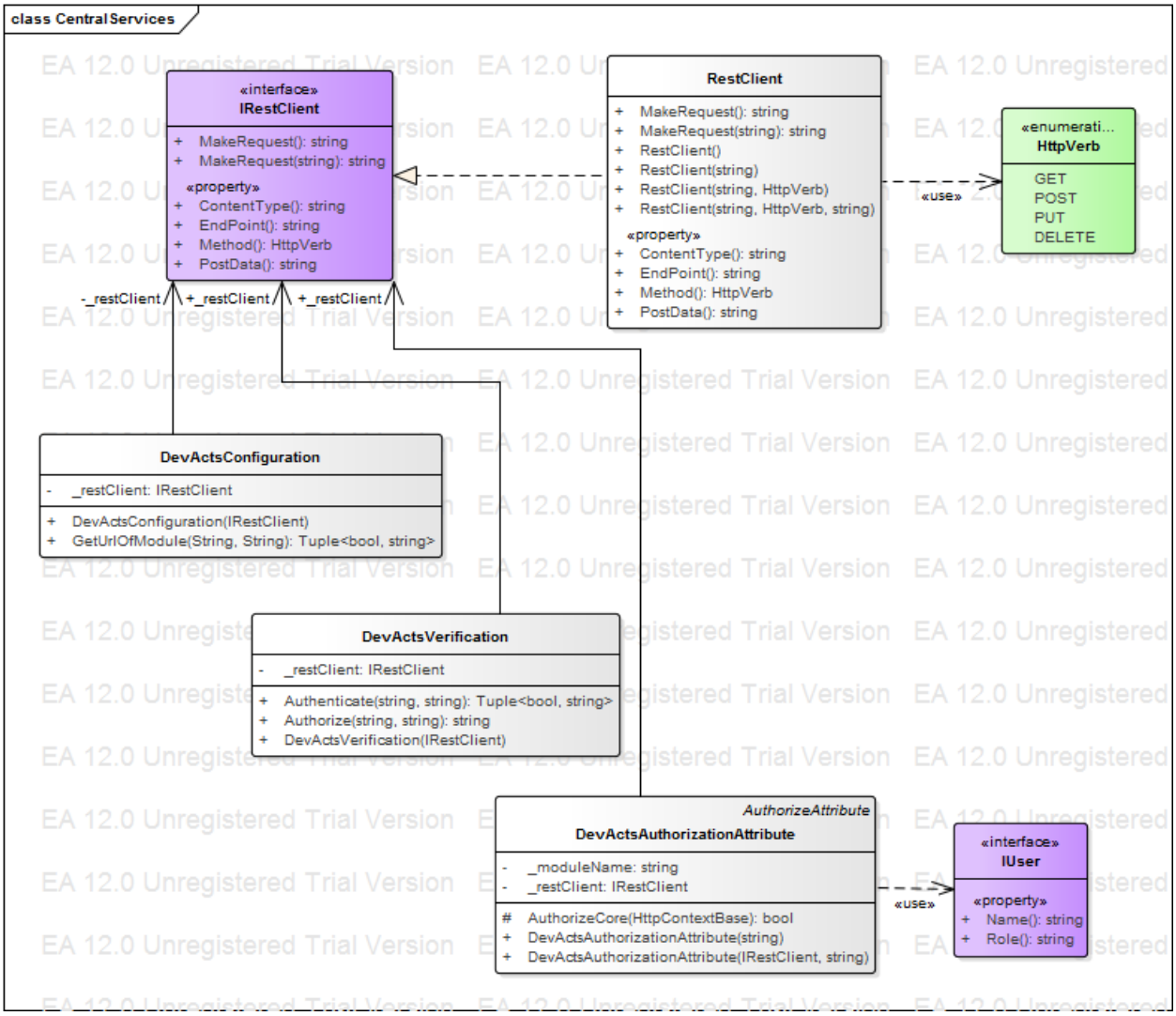
Obrázok 3.1: Návrh metódy pre posielanie logov

3.2 Návrh Core.CentralServices

Pri Core.CentralServices nebolo potrebné navrhovať nové funkcie systému ani žiadne iné mechanizmy, úlohou bol iba presun autorizácie a autentifikácie do projektu Core.CentralServices, tak aby mohli byť referencované ostatnými modulmi. Muselo však dôjsť k drobným zmenám autorizácie, tak aby mohla byť využitá vo viacerých moduloch. Došlo k nasledujúcim úpravám: *DevActsAuthorizationAttribute* dostáva ako parameter v konštruktore názov modulu, pre ktorý má prebiehať autorizácia. Taktiež musela byť upravená trieda *AuthorizeCore*, tak aby dokázala pracovať s rôznymi typmi používateľov naprieč rôznymi modulmi. Za týmto účelom vzniklo rozhranie IUser, ktoré má iba dve vlastnosti: meno a rola používateľa. Potom, každý modul, ktorý využíva túto metódu na autorizáciu musí do Session priradzovať inštanciu triedy, ktorá dedí od IUser rozhrania. Mala by sa pridať trieda *DevActsConfiguration*, ktorá bude implementovať volanie služby na získanie URL modulu, podľa názvu modulu a kľúčového slova. Na obrázku 3.2 sa nachádza diagram tried zabezpečujúcich autentifikáciu a autorizáciu.

3.3 Návrh Core.CentralServices.Config

Na to aby bolo jednoduché získavať konfigurčné hodnoty pre jednotlivé moduly z jedného miesta, navrhujem v knižnici Core.CentralServices vytvoriť časť, ktorá sa bude zaoberať čisto získavaním konfiguračných hodnôt z Administrčného portálu, pomocou volania REST služby. Taktiež sa navrhlo vytvorenie generickej triedy, od ktorej by dedili triedy v iných moduloch a mali by na starosti správu konfguračných hodnôt. Ako poslednú vec navrhujeme zadefinovanie globálneho enumu pre názvy všetkých používaných modulov, aby sa predišlo používaniu surových stringov v kódoch iných modulov a tým sa znížila aj šanca na chyby.



Obrázok 3.2 Diagram tried potrebných pre autorizáciu a autentifikáciu.

4 Implementácia

Implementácia navrhnutej funkcionality bola realizovaná podľa návrhu v nasledujúcej kapitole bude detailnejšie opísané niektoré dôležité implementačné detaily a rozhodnutia, ktoré k nim viedli.

4.1 Implementácia posielania logov

Implementácia prebehla podľa návrhu. Do každej metódy, ktorá logovala správy sa pridala jedna z metód:

```
private async void SendLog(string message, string type, string module, Logger mLogger)
```

```
private async void SendLog(string message, string type, string module, Logger mLogger, params object[] args)
```

Prvá metóda slúži na posielanie logov, ktoré majú len správu a druhá na posielanie logov, ktoré obsahujú aj pole objektov.

Pri posielaní logov sa adresa Administračného portálu určuje z konfiguračného súboru. V konfiguračnom súbore sa taktiež nachádza premenná, ktorá určuje či sa má posielať správa cez HTTP alebo HTTPS protokol.

Poslanie logov je úspešné, ak je odpoveď „Created“. V inom prípade sa zapíše log do lokálneho súboru o chybe spojenia s Administračným portálom. Implementácia autentifikácie služby prebehla podľa návrhu. Autentifikačná pipeline na strane administračného portálu je opísaná v dokumentácii administračného portálu:

https://onedrive.live.com/redir?resid=957CC12FD13AF109!5439&authkey=!ACtdRURSlw8uB_Y&ithint=file%2cdocx

4.2 Implementácia Core.CentralServices

Implementácia prebehla tak, ako bolo opísané v návrhu. Pri presune tried zabezpečujúcich logovanie došlo k menším komplikáciám, kde presunuté triedy vyžadovali referencovanie projektu Core.Logging, čím by vznikla cyklická závislosť medzi dll, ktorá v .Net nie je povolená, preto muselo dôjsť k menšiemu refaktoringu týchto tried, resp. metód. Implementácia autorizácie a autentifikácie prebehla tak ako bolo navrhnuté. Na obrázku 3.3 sa nachádza metóda *AuthorizeCore* ktorá je jadrom autorizácie. Pri autorizácii sa využíva mechanizmus frameworku .Net nazývaný bezpečnosť založená na rolách (angl. role-based security). Spolu s deklaratívnym riadením bezpečnosti pomocou atribútov.

```
protected override bool AuthorizeCore(HttpContextBase httpContext)
{
    if (httpContext.User.Identity.IsAuthenticated)
    {
        var user = httpContext.Session["User"] as IUser;

        if (user == null) return false;

        // Always use rest service for authorization
        if (_restClient != null)
        {
            _restClient.EndPoint = Settings.Default.DevActsEndpoint + "api/users/authorize";
            _restClient.PostData = new JavaScriptSerializer()
                .Serialize(new AuthorizationRequest()
                {
                    UserName = user.Name,
                    Module = _moduleName
                });
            _restClient.Method = HttpVerb.POST;
            _restClient.ContentType = "application/json; charset=UTF-8";
        }
        else
        {
            return false;
        }

        var authorizationResult = _restClient.MakeRequest(Encoding.UTF8);
        var authorizationResponse = new JavaScriptSerializer().Deserialize<AuthorizationResponse>(authorizationResult);

        string userRole = authorizationResponse.Role;

        if (userRole == null) return false;

        // if user role changed update Session["User"] with new role
        if (userRole != user.Role)
        {
            user.Role = userRole;
            httpContext.Session["User"] = user;
        }

        return Roles.Split(',').Contains(userRole);
    }
    return false;
}
```

Obrázok 3.3 Implementácia kľúčovej metódy AuthorizeCore

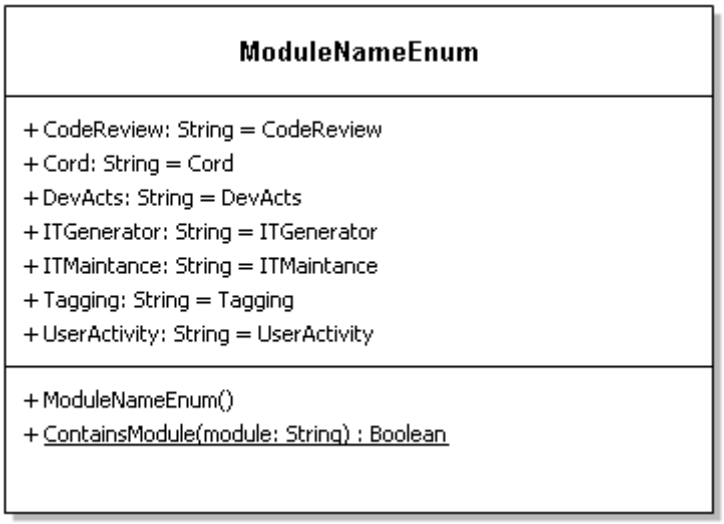
Trieda *DevActsConfiguration* bola implementovaná tak, ako bola navrhnutá. Metóda `GetUrlOfModule` má na vstupe dva parametre:

- Názov modulu taký, aký je uvedení v konfigurácii systému DevActs
- Kľúč k URL (momentálne existuje len baseUrl, ktorý je defaultne nastavený ako parameter)

Názvy modulov sú uložené v properties projektu, aby mohli byť ľahko konfigurovateľné.

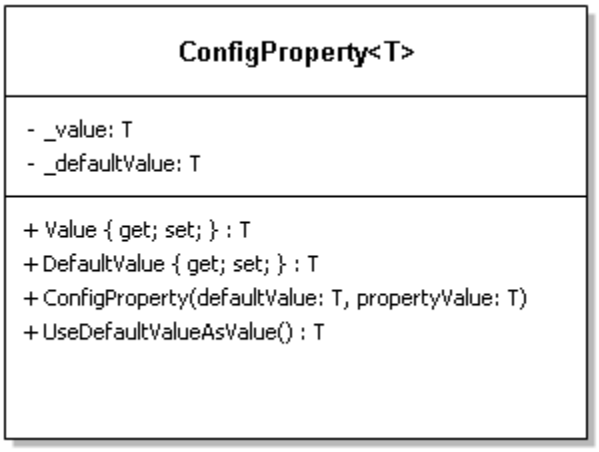
4.3 Implementácia Core.CentralServices.Config

Implementácia globálneho enumu pre názvy modulov bola implementovaná ako samostatná trieda s verejnými konštantnými stringami. Do tejto triedy bola pridaná aj statická metóda na zistenie, či názov stringu je zadefinovaný ako názov v modulu v tomto enume. Táto metóda má využitie v prípade, keď chceme validovať názvy modulov, ktoré sa vkladajú do databáz a v prípade vloženia zlej hodnoty by bola nutná migrácia na nápravu. Túto triedu môžeme vidie na obrázku č. 3.4.



Obrázok 3.4 Class diagram triedy ModulNameEnum.

Na to aby sme v prípade zlej nastavenej konfiguračnej hodnoty vedeli použiť defaultnú hodnotu, ktorá by mala byť vždy správna, tak sme implementovali samostatnú triedu ConfigProperty. Táto tried je iba akoby obaľovacia trieda, ktorá združuje dve hodnoty (aktuálnu hodnotu + defaultnú hodnotu). Keďže je dobré určiť typ pre každú konfiguračnú hodnotu, je táto trieda templatová aby sa mohla použiť pre akýkoľvek dátovy typ potrebujeme. Jej implementáciu môžeme vidieť na obrázku č. 3.5.

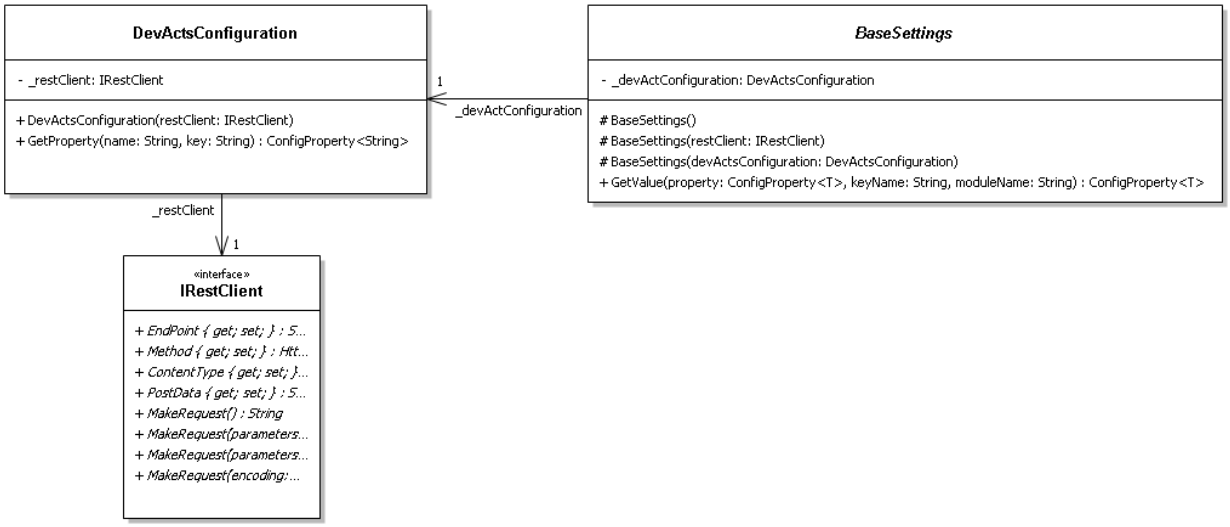


Obrázok 3.5 Implementácia triedy pre konfiguračné hodnoty.

Konfiguračná trieda je abstraktná a určená ako base trieda na oddedenie pre rôzne moduly, ktoré si už túto triedu rozšíria o potrebnú funkcionalitu. Základnou vlastnosťou tejto triedy je generické získavanie a pretypovanie na potrebný dátový typ konfiguračných hodnôt z administračného portálu. Na to aby sa moduly pri každom potrebnom dopytovaní na konfiguračné hodnoty nemuseli dopytovať na admin. Portál pomocou HTTTP a REST služby, je vhodné ma túto konfigurčnú hodnotu ako nullable a dopytovať sa na jej hodnotu len prvý krát, keď je jej hodnota null. Takto dokážeme ušetriť odozvu ako aj uľahčenie dátovej trafiky.

Na volanie REST služby pre kniguračné hodnoty sa taktiež vytvorila trieda DevActsConfiguratiom, ktorá sa pomocou IRestClient rozhrania dopytuje na REST službu umiestnenú na administračnom portále, podľa kľúča hodnoty a názvu modulu. V prípade úspechu vráti danú konfiguračnú hodnotu, inak vyhodí aplikačnú výnimku so správou, prečo sa nepodarilo získať danú hodnotu.

Celkové fungovanie získavania hodnoty pomocou týchto tried môžeme vidieť na obrázku č 3.6.



Obrázok 3.6 Diagram tried pre zobrazenie fungovania systému na získavanie konfiguračných hodnôt.

5 Testovanie

Testovanie prebehlo štandardne, sú implementované jednotkové testy pre:

- metódu vytvorenia logu
- metódu na posielanie logov do Administračného portálu
- autorizáciu a to aj pre úspešný aj neúspešný scenár
- metódu na získavanie knfiguračnej hodnoty ako v prípade úspechu, tak aj neúspechu.