Eötvös Loránd Tudományegyetem
Informatikai Kar

Webes alkalmazások fejlesztése

14. fejezet

Szolgáltatások adatkezelése (WCF)

Giachetta Roberto

A jegyzet az ELTE Informatikai Karának 2016. évi jegyzetpályázatának támogatásával készült

Szolgáltatások adatkezelése

Adatmennyiség

- Adatszolgáltatók kérésenként tetszőleges mennyiségű adatot fogadhatnak/küldhetnek
  - a konfiguráció határozza meg az adatmennyiség felső korlátait, amik alapértelmezetten kis mennyiségek (a szerver leterheltségének és a túlterheléses támadások korlátozására)
  - az adatmennyiség különböző szinteken szabályozott: *webszerver* (maximális kérés mérete HTTP alapú adatközlésnél), *üzenet* (teljes üzenet hossza), illetve *tartalom* (egy bejegyzés/érték hossza)
  - a korlátozás szerver és kliens oldalon is definiált

Szolgáltatások adatkezelése

Adatmennyiség

- A maximális adatmennyiséget számos módon szabályozhatjuk a konfigurációban, pl.:
  - fogadható kérés mérete (szerver): **`HttpRuntime.MaxRequestLength`**
  - fogadható üzenet mérete (szerver és kliens): **`Binding.MaxReceivedMessageSize`**
  - fogadható bináris adatmennyiség értékenként (szerver és kliens): **`Binding.ReaderQuotas.MaxArrayLength`**
  - fogadható szöveges adatmennyiség értékenként (szerver és kliens): **`Binding.ReaderQuotas.MaxStringContentLength`**

Szolgáltatások adatkezelése

Adatkommunikáció módja

- Az adatkommunikáció többféle módon bonyolítható le (**`Binding.TransferMode`**):
  - csomag alapon (**`Buffered`**): az adatokat puffer tömbökbe gyűjti, és együttesen továbbítja (alapértelmezett)
    - ekkor a puffer méretét is szabályozhatjuk (**`Binding.MaxBufferSize`**)
  - folyam alapon (**`Streamed`**): az adatokat kisebb darabokban, folyamatosan továbbítja
  - vegyes (**`StreamedRequest`**, **`StreamedResponse`**)
- Nagyobb adatmennyiségnél célszerű folyam alapú kommunikációt használni

Szolgáltatások adatkezelése

Adatmennyiség

- Pl.:

…

```
<basicHttpBinding>
   <binding transferMode="Streamed"
            maxReceivedMessageSize="268435456">
   <!-- az adatközlés adatfolyam alapú -->
      <readerQuotas maxDepth="32"
                    maxStringContentLength="8192"
                    maxArrayLength="268435456"
                    maxBytesPerRead="4096"
                    maxNameTableCharCount="16384" />
   <!-- átállítjuk a maximum üzenet és tömb
        méretet a nagyobb adattartalomhoz -->
```

…

Szolgáltatások adatkezelése

Példa

Feladat: Készítsünk egy adminisztrációs alkalmazást az utazási ügynökség számára, ahol az egyes kiadó épületekhez tartozó képeket tudják kezelni.

- az adatokat adatbázisban (`TravelAgency`) tároljuk, beleértve a képeket (byte-tömb formában), amelyekből két méretet (kisebb, nagyobb) tárolunk el és használunk fel a későbbiekben
- egy épülethez több kép is tartozhat, így külön táblába helyezzük (`BuildingImage`)
- az adatbázist entitásmodellel építjük fel a szerveren, amely továbbítja az entitásokat a kliensnek (amelyek automatikusan adatszerződések lesznek)

Szolgáltatások adatkezelése

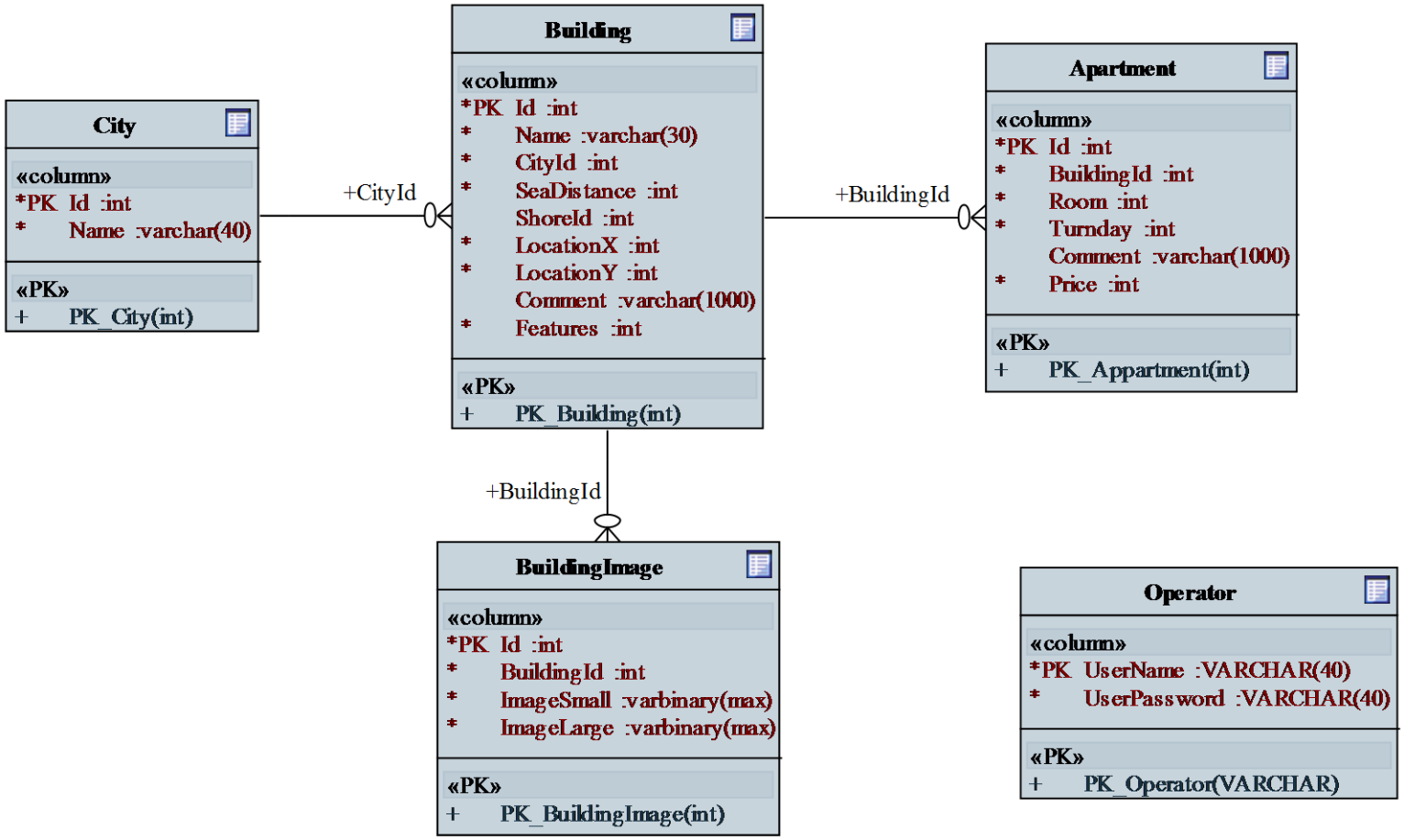
Példa

- a felhasználót (**`Operator`**) név/jelszó alapján azonosítjuk
- megemeljük kellőképpen az adatmennyiség korlátokat és használjuk folyam alapú kommunikációt
- kliens oldalon a képet töltsük be (**`BitmapImage`**), alakítsuk át PNG formátumra és méretezzük át (**`PngBitmapEncoder`**), majd töltsük át byte-tömbbe a tároláshoz (**`VARBINARY`** formátumban) és továbbításhoz
- a továbbítást azonnal elvégezzük, majd frissítjük az adatokat a szerverről
- a megjelenítéshez megfelelő átalakító kell (**`BuildingImageConverter`**), amely a byte-tömböt képpé alakítja

Szolgáltatások adatkezelése

Példa

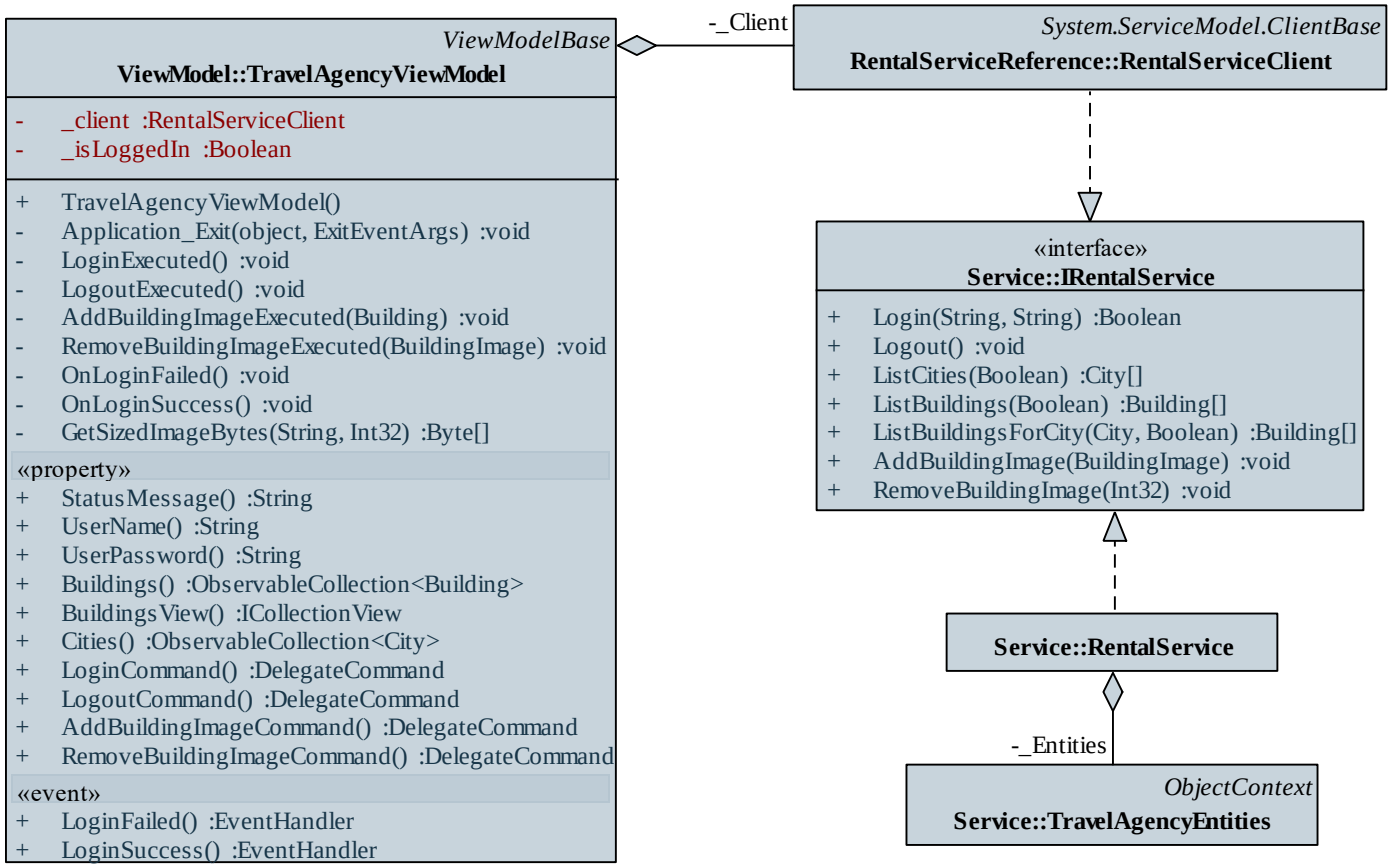
Tervezés (adatbázis):



Szolgáltatások adatkezelése

Példa

Tervezés (kliens és szerver):



Szolgáltatások adatkezelése

Példa

Megvalósítás (`TravelAgencyViewModel.cs`*):*

```
private void AddBuildingImageExecuted(Building
                                      building){
    …
    BuildingImage image = new BuildingImage{
       BuildingId = building.Id,
       ImageSmall =
          GetSizedImageBytes(dialog.FileName, 100),
          // az eredeti képet átméretezzük
       … }; // létrehozzuk a képet

    _Client.AddBuildingImage(image);
       // felvesszük a képek közé a szerveren
    …
```

Szolgáltatások adatkezelése

Példa

Megvalósítás (`TravelAgencyViewModel.cs`*):*

```
// kép átméretezése és konvertálása
private Byte[] GetSizedImageBytes(String path,
                                  Int32 height){
    BitmapImage image = new BitmapImage();
    image.BeginInit(); // kép betöltése
    image.UriSource = new Uri(path);
    image.DecodePixelHeight = height;
       // megadott méretre
    image.EndInit();

    PngBitmapEncoder encoder =
       new PngBitmapEncoder();
       // átalakítás PNG formátumra
```

Szolgáltatások adatkezelése

Példa

Megvalósítás (`TravelAgencyViewModel.cs`*):*

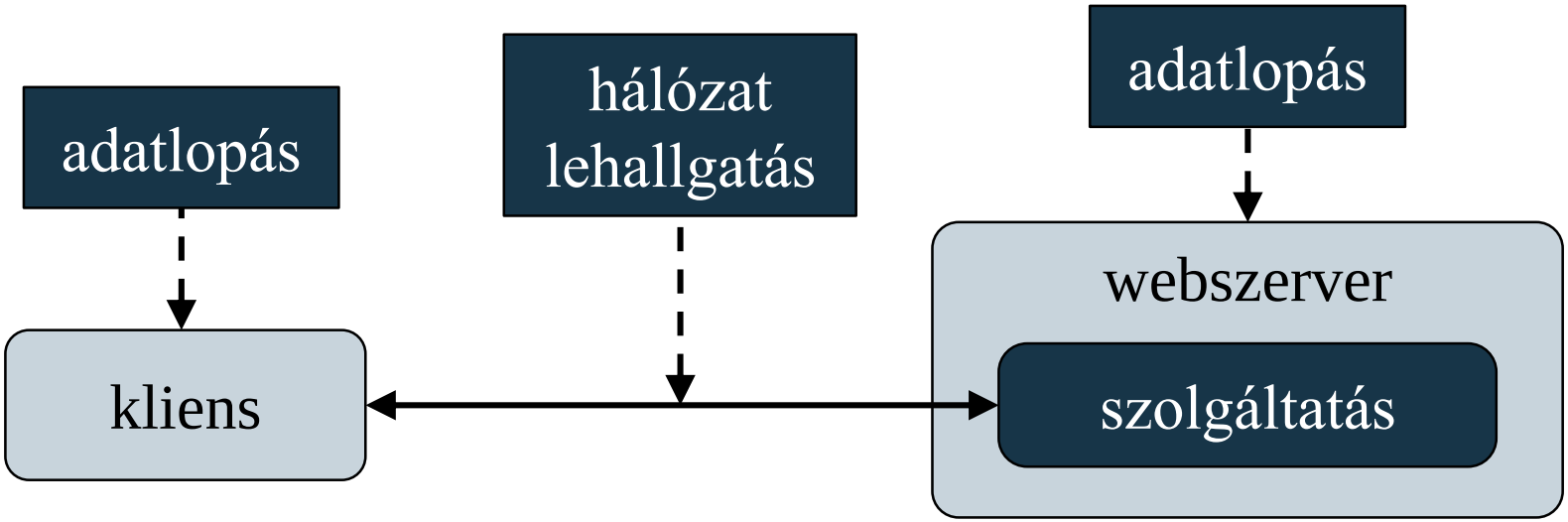
```
        encoder.Frames.Add(BitmapFrame.Create(image));

        using (MemoryStream stream =
                new MemoryStream())
            // átalakítás byte-tömbre
        {
            encoder.Save(stream);
            return stream.ToArray();
        }
    }
```

Szolgáltatások adatkezelése

Biztonsági kérdések

- Amennyiben a szolgáltatás az interneten elérhető, számos ponton megtámadhatóvá válik, pl.:
  - lehallgathatják a hálózati üzeneteket
  - megtámadhatják a klienst/szervert, és kilophatják a tárolt adatokat (pl. memóriából, adatbázisból)



Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- A *hálózat lehallgatás* során a küldött/fogadott adatok kerülhetnek idegen kezekbe, amelyek egyszerű protokollok (pl. SOAP) használata miatt könnyen kiolvashatóak
  - az üzeneteket megfelelően kell titkosítani, erre szolgál a *Transport Layer Security* (*TLS*, *SSL*)
- A WCF két szinten támogatja a biztonságos kapcsolatokat:
  - *kapcsolat*: minden üzenetet külön kódolni és dekódolni kell, amihez egy *tanúsítvány* (*certificate*) kerül megosztásra a kliens számára
  - *üzenet*: minden üzenetnek külön tanúsítványa van (önhordozó), és csak a tartalma kerül kódolásra

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- A titkosításhoz tartozik szorosan a felhasználói azonosítás, hiszen eleve csak megbízható kliensekkel akarjuk a titkosított csatornát megosztani
  - több megoldás támogatott, pl. Windows authentikáció
- Egyedi felhasználónév/jelszó azonosítást is definiálhatunk a **`UserNamePasswordValidator`** osztály specializációjával
  - a **`Validate`** metódusban tetszőleges ellenőrzéseket valósíthatunk meg felhasználónévre, jelszóra
  - ha a metódus kivételt dob (**`SecurityTokenException`**), akkor az ellenőrzés érvénytelen, egyébként érvényes
  - a konkrét kivételt a kliens nem kapja meg

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- Pl.:

```
public class EmployeeValidator :
    UserNamePasswordValidator // ellenőrző osztály
{
    public override void Validate(String userName,
                                  String password){
        if (String.IsNullOrEmpty(userName))
            throw new SecurityTokenException();
        // ha nem adott meg felhasználónevet,
        // kivételt dobunk

        … // további ellenőrzések
    }
}
```

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- A szolgáltatás viselkedésénél konfiguráljuk az azonosítás módját (**`ServiceBehavior.ServiceCredentials`**)
  - egyedi (**`Custom`**) azonosítás esetén az azonosításhoz tartozó típusa típus esetén megadjuk a teljes elérési útvonalat (névtér, típus, szerelvény)
- A TSL használatához a webszervernek szolgáltatnia kell egy X.509 tanúsítványt, amely tartalmazza a hozzáférési kulcsokat
  - a tanúsítvány vásárolható, vagy készíthető (pl. *SelfCert*), utóbbi esetben nem lesz hiteles
  - a tanúsítványhoz tartozik egy azonosító, amely általában a szerver azonosítója (névtere)

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- Pl.:

```
<serviceCredentials>
    <!-- A szolgáltatás biztonsági adatai -->
    <userNameAuthentication
        userNamePasswordValidationMode="Custom"
        customUserNamePasswordValidatorType=
          "EmployeeService.EmployeeValidator,
           EmployeeService" />
        <!-- egyedi azonosítás, amely a
             megadott típust fogja használni a
             név/jelszó ellenőrzésre a megadott
             szolgáltatáshoz -->
    …
```

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- Pl.:

```
<serviceCredentials>
    …
    <serviceCertificate findValue="myDomain"
       storeLocation="LocalMachine"
       storeName="My"
       x509FindType="FindBySubjectName"/>
    <!- tanúsítvány (szerver) azonosítója,
        helye (tároló és mappa), és a keresés
        módja -->
    …
```

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- A kötésnek megadható, hogy milyen biztonsági üzemmódot (**`Security`**) használjon, ezt követően az üzenetek biztonságos csatornán keresztül továbbítódnak
  - csak egyes kötéssekkel használható (pl. **`WsHttpBinding`**)
  - pl.:

```
<wsHttpBinding>
 <binding>
   <security mode="Message">
      <message clientCredentialType="UserName"/>
   </security>
   <!- üzenet alapú titkosítás, ami név és
       jelszó alapján azonosít -->
   …
```

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- A kliensnél meg kell adnunk az azonosítás paramétereit
  - kliens oldalon is a **`ClientCredentials`** tulajdonság tartalmazza a biztonsági beállításokat
    - a **`UserName`** írja le a felhasználó azonosítását, ennek meg kell adnunk a felhasználónevet, illetve jelszót (**`UserName`**, **`Password`**)
    - megadhatjuk a tanúsítvány hitelesítésének módját (**`CertificateValidationMode`**), így elfogadhatjuk nem hiteles tanúsítványokat is (**`None`**)
  - a konfigurációban megadjuk a szerver (tanúsítvány) azonosítóját (**`Endpoint.Identity.Dns`**)

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- Pl.:

```
… // kliens példányosítása

client.ClientCredentials.ServiceCertificate.
   Authentication.CertificateValidationMode =
   X509CertificateValidationMode.None;
   // nem várjuk el a megbízható tanúsítványt

// beállítjuk a felhasználónevet és jelszót
client.ClientCredentials.UserName.UserName = …;
client.ClientCredentials.UserName.Password = …;

… // hívhatjuk az első műveletet
```

Szolgáltatások adatkezelése

Üzenet titkosítás és azonosítás

- Pl.:

```
<client>
    <endpoint …>
        <identity>
            <dns value="myDomain" />
            <!-- megadjuk a szerver (tanúsítvány)
                 azonosítóját --->
        </identity>
    …
```

- A kliens is szolgáltathat tanúsítványt, erre vonatkozó feltételeket a kliens és a szerver oldalon is meg kell adnunk (`clientCertificate`)

Szolgáltatások adatkezelése

Példa

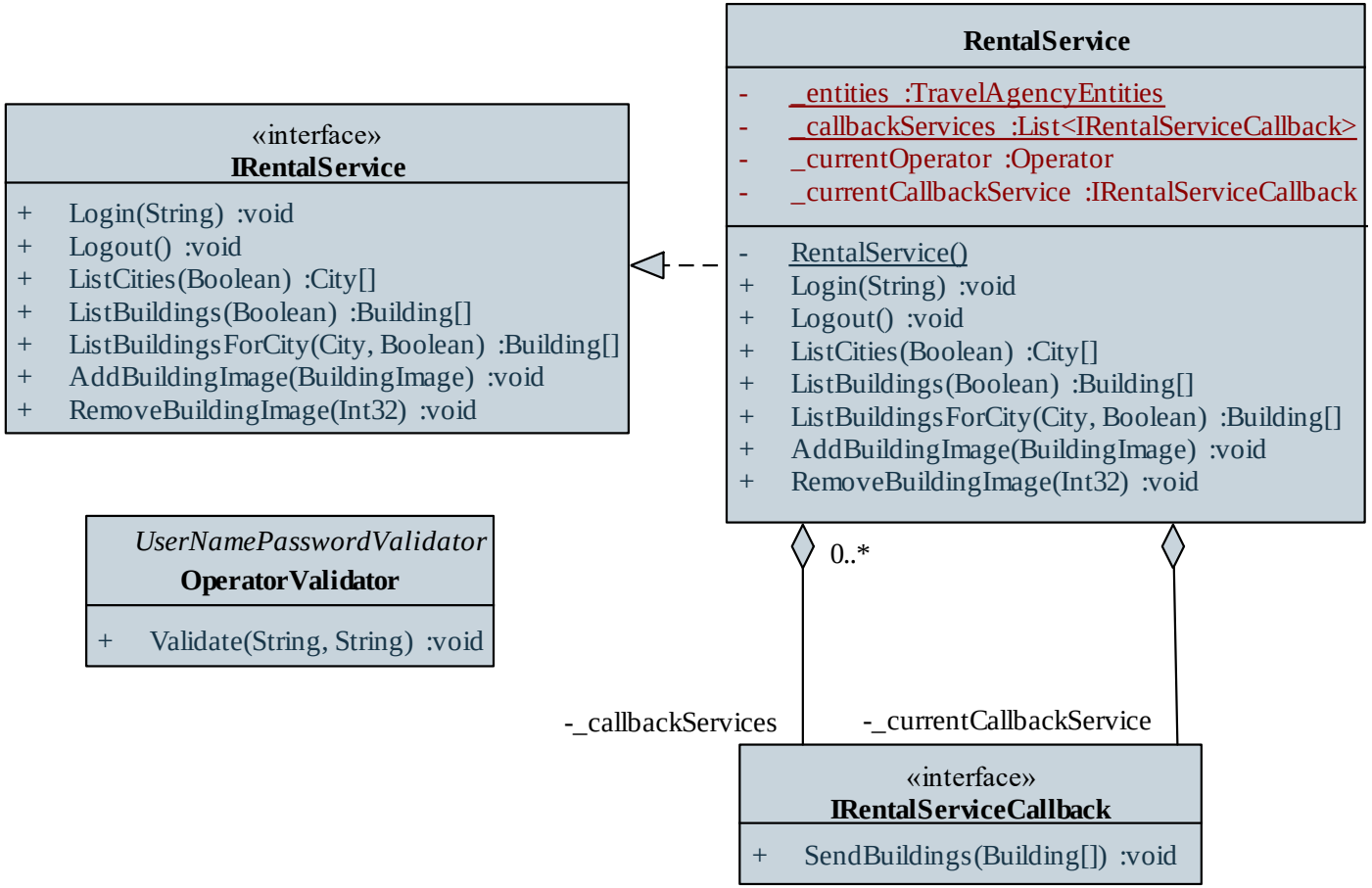
Feladat: Módosítsuk az előző adminisztrációs rendszert úgy, hogy biztonságos és hatékony legyen.

- a kommunikációt TLS segítségével valósítjuk meg, név/jelszó alapú azonosítással (az adatokat az adatbázisból hitelesítjük), ezzel megfelelő azonosító típust (`OperatorValidator`) hozunk létre
- az adatbázisban a jelszót binárisan, SHA1 kódolva tároljuk
- a kliens a jelszót `PasswordBox` segítségével kéri be
- a szolgáltatást munkafolyamat alapon példányosítjuk és visszahívások segítségével jelezzük az adatbázisbeli változásokat, ehhez szükséges visszahívó interfészt (`IRentalServiceCallback`) és típust hozunk létre

Szolgáltatások adatkezelése

Példa

Tervezés:



Szolgáltatások adatkezelése

Példa

Megvalósítás (`OperatorValidator.cs`*):*

```
public override void Validate(String userName,
                              String password){
    …
    Byte[] passwordBytes;

    using (SHA1CryptoServiceProvider provider =
           new SHA1CryptoServiceProvider()) {
       // SHA1 kódoló, mivel a jelszavakat kódoltan
       // tároljuk
       passwordBytes = provider.ComputeHash(
          Encoding.UTF8.GetBytes(password));
          // átalakítjuk a jelszót kódoltra
    }
```

Szolgáltatások adatkezelése

Példa

Megvalósítás (`OperatorValidator.cs`*):*

```
      Operator currentOperator = entities.Operator.
       FirstOrDefault(op => op.UserName == userName);
       // lekérdezzük a felhasználót az adatbázisból

       if (currentOperator == null ||
          !passwordBytes.SequenceEqual(
           currentOperator.UserPassword))
         // ha nem találtuk meg, vagy a jelszó nem
         // egyezik, kivételt dobunk
         throw new SecurityTokenException("Username
                          or password does not match.");
       …
  }
```