Refaktoring

Rename a variable with a clearer or more informative name

U početku je bila ideja da agencija bira više slika za navedenu destinaciju, što je na kraju promijenjeno, pa je potrebno preimenovati varijablu. Također i naziv enum tipa bi trebao biti u jednini pa je također ispravljeno.

```
public class Destinacija
{
    private int id;
    private String naziv;
    private String drzava;
    private Kontinenti kontinent;
    private Image slikeDestinacije;
```

Slika prije refaktoringa

```
public class Destinacija
{
    private int id;
    private String naziv;
    private String drzava;
    private Kontinent kontinent;
    private Image slikaDestinacije;
```

Slika poslije refaktoringa

Create and use null objects instead of testing for null values

Ovdje smo postavili sliku na null, jer agencija može da ne izabere nikakvu sliku za destinaciju. Bolje je postaviti na null ako ne pošalje nikakav Image nego provjeravati da li je null.

```
public Destinacija(string naziv, String drzava, Kontinent kontinent, Image slikaDestinacije = null)
{
    id = Globalna.idSvihDestinacija++;
    Naziv = naziv;
    Drzava = drzava;
    Kontinent = kontinent;
    SlikaDestinacije = slikaDestinacije;
}
```

Global variables are used

```
public static class Globalna
{
    public static TravelBook nasaAgencija = new TravelBook();
    public static int prijavljenaAgencijaId = 0;
    public static bool trenutnaPutovanja = true;
    public static int idSvihAgencija = 0;
    public static int idSvihDestinacija = 0;
    public static int idSvihPutovanja = 0;
    public static int idSvihKartica = 0;
    public static int idSvihHotela = 0;
    public static int idSvihPrevoza = 0;
    public static int idSvihKorisnika = 0;
}
```

Ovdje koristimo static klasu koja je globalna i njene varijable. TravelBook je kontejner klasa (mora biti samo jednom instancirana inače bi se podaci izgubili) za sva putovanja, korisnike, destinacije ...

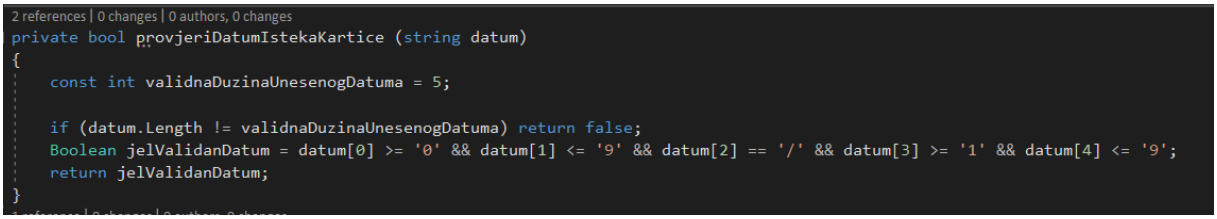
```
private void Dugme_RegistrujSe(object sender, RoutedEventArgs e)
{
    if (validacijaPodataka())
    {
        Kartica nova = new Kartica((VrstaKartice)tTipKartice.SelectedItem, tDatumIsteka.Text, tBrojKartice.Text, Convert.ToInt32(tCSC.Text));
        Globalna.nasaAgencija.Kartice.Add(nova);
        KarticaAzure kart = new KarticaAzure();
        kart.dodajKarticu(nova);
        bool jesuIsteSifre = tSifra.Password.ToString().Equals(tSifraPonovo.Password.ToString());
        if (jesuIsteSifre)
        {
            Agencija agencija = new Agencija(tNaziv.Text, nova, tTelefon.Text, tMail.Text, tGrad.Text, tAdresa.Text, tSifra.Password.ToString
            r.registrujAgneciju(agencija);
            try
            {
                AgencijaAzure agencijaAzure = new AgencijaAzure();
                agencijaAzure.dodajAgenciju(agencija);
                var dialog = new MessageDialog("Uspješno ste registrovali agenciju!");
                dialog.ShowAsync();
            }
            catch (Exception ex)
            {
                MessageDialog msgDialogError = new MessageDialog("Error : " + ex.ToString());
                msgDialogError.ShowAsync();
            }
            Frame.Navigate(typeof(Prijava));
        }
        else
        {
            r.Poruka = new MessageDialog("Pogrešna šifra! Pokušajte ponovno.");
            r.Poruka.ShowAsync();
        }
    }
```

Na slici iznad je primjenjen refaktoring **Extract a routine** gdje pozivamo metodu validacijaPodataka() u kojoj se nalazi čitav kod za validaciju, koji je prije bio u metodi Dugme_RegistrujSe.
Dugme_RegistrujSe je također preimenovano u View i u .cs dijelu iz Button_Click1, pošto imamo 2 buttona te iz naziva metode se nije odma moglo zaključiti o kojem se radi, ovako naziv daje više informacija.
Varijabla jesuIsteSifre je dodana kako bi primjenili refaktoring **Decompose a boolean expression** i također da bi se bolje snašli u kodu.
Pass a whole object rather than specific fields je primjenjeno kod metode registrujAgenciju koja prije imala previse parametara (**A parameter list has too many parameters**) . Objekat agencije je

kreiran tek u metodi, pa je prepravljena da prima čitav objekat. Time što smo odma objekat kreirali uštedili smo vrijeme kod upisa u bazu, jer smo tu prije refaktoringa morali ponovo pristupat objektu pirstupajuci zadnjem iz liste.

Data members are public

Svi atributi u klasama su private, za pristup se koriste geteri i seteri.



```
2 references | 0 changes | 0 authors, 0 changes
private bool provjeriDatumIstekaKartice (string datum)
{
    const int validnaDuzinaUnesenogDatuma = 5;

    if (datum.Length != validnaDuzinaUnesenogDatuma) return false;
    Boolean jelValidanDatum = datum[0] >= '0' && datum[1] <= '9' && datum[2] == '/' && datum[3] >= '1' && datum[4] <= '9';
    return jelValidanDatum;
}
```

Metoda je preimenovana tako da bi davala više informacija o svom radu. Također imamo validnaDuzinaUnesenogDatuma koja zadovoljava **Replace a magic number with a named constant** . Čitava provjera datuma je dodijeljena varijabli jelValidanDatum, te nema nikakvih if then else iskaza. Te možemo reci da odma vraća vrijednost tj. ispunjava **Return as soon as you know the answer instead of assigning a return value within nested if-then-else statements .** Metoda se poziva dvaput tako da smo izbjegli dupliciranje koda(**Code is duplicated**).

Pass specific fields rather than a whole object je iskorišten kod funkcije pošaljiEmail, jer nam nisu potrebne nikakve informacije osim imena korisnika, naziva destinacije i id-a putovanja, pa ne šaljemo čitave objekte.

```
n("PosaljiEmail", "Home",new { destinacija = @Html.Action("vratiIme", "DestinacijaAzures", new { id= item.idDestinacije }), ime = User.Identity.GetUserName(), idPutovanja = item.id })'
```