Razvoj payment gateway-a za paypal express checkout servis

Turopoli, Dino

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij matematike i računarstva

Dino Turopoli

Razvoj payment gateway-a za PayPal Express Checkout servis

Diplomski rad

Osijek, 2018.

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij matematike i računarstva

Dino Turopoli

Razvoj payment gateway-a za PayPal Express Checkout servis

Diplomski rad

Mentor: izv. prof. dr. sc. Domagoj Matijević

Osijek, 2018.

Sadržaj

1 Uvod

Ovaj diplomski rad sastoji se od praktičnog i teorijskog dijela. U teorijskom dijelom razraditi će se teorija vezana za e-trgovine i elektroničko plaćanje, dok će se u praktičnom dijelu rada izrađivati prototip e-trgovine kako bi se mogao implementirati usmjerivač plaćanja (eng. payment gateway) za PayPal Express Checkout servis. Web aplikacija imati će osnovne funkcije e-trgovine poput dodavanja, uređivanja i kupovine proizvoda. Plaćanje će se odvijati putem PayPal-a koristeći PayPal Express Checkout servis.

U prvom poglavlju proći ćemo kroz teoriju vezanu za e-trgovinu, objasniti različite modele e-trgovina i načine elektroničkog plaćanja.

Drugo poglavlje vezano je za model plaćanja kreditnim karticama. Kako se praktični dio rada bazira na plaćanju kreditnim karticama u ovom poglavlju ćemo detaljno opisati teoriju vezanu za proces plaćanja karticama i razraditi problematiku usmjerivača plaćanja, načine implementiranja usmjerivača plaćanja te PCI DSS sigurnosni standard.

Zadnje poglavlje pokriva praktični dio rada i podijeljeno je na dva poglavlja. U prvom dijelu sažeto ćemo se upoznati sa alatima i tehnologijama korištenim za izradu same aplikacije. U drugom dijelu najprije ćemo se upoznati s osnovnim funkcijama aplikacije. Nakon toga baviti ćemo se osposobljavanjem PayPal Developer računa kako bi smo mogli testirati aplikaciju i na kraju ćemo razviti usmjerivač plaćanja prema PayPal Express Checkout servisu te testirati samu aplikaciju pomoću PayPal sandbox okruženja.

2 Elektronička trgovina

Elektronička trgovina (eng. e-commerce) je oblik trgovine koji omogućuje korisnicima obavljanje kupovine iz svog doma putem interneta. E-trgovina predstavlja najprofitabilniji oblik trgovine jer pospješuje kvalitetu usluga i ubrzava proces kupnje i dostave proizvoda. Neka od bitnijih karakteristika e-trgovine su:

- **Bezgotovinsko plaćanje** - omogućuje kupcu da obavi kupovinu putem kreditnih i debitnih kartica i drugih oblika elektroničkog plaćanja bez korištenja gotovine,
- **Dostupnost** – sve e-trgovine dostupne su korisnicima 24 sata na dan i svaki dan u godini. Sve što korisniku treba da bi pristupio e-trgovini je internet,
- **Marketing** – korištenjem e-trgovine trgovac može lako širiti svoje tržište na globalnoj razini bez prevelikih ulaganja,
- **Korisnička podrška** – e-trgovina pruže korisniku podršku prije, tijekom i nakon kupovine proizvoda putem raznih servisa,
- **Upravljanje skladištem** – e-trgovina automatizira upravljanje skladištem. Narudžbe i računi se instantno kreiraju i odmah se iz baze podataka brišu prodani proizvodi i na taj način upravljanje skladištem postajem efikasnije i jednostavnije.

Postoje različiti *poslovni modeli* (eng. business models) e-trgovine. Kategoriziraju se na osnovu korisnika koji međusobno posluju. Postoje sljedeći modeli:

- **poduzeće-poduzeće** (eng. business – to – business) – web stranica prodaje proizvod posredujećem poduzeću koje onda taj proizvod prodaje krajnjem potrošaču,
- **poduzeće-potrošač** (eng. business – to – consumer) – web stranica direktno prodaje proizvod krajnjem potrošaću,
- **potrošač-potrošač** (eng. consumer – to – consumer) – web stranica pomaže korisniku prodati njegov proizvod,
- **potrošač-poduzeće** (eng. consumer – to – business) – korisnik dolazi na web stranicu koja prikazuje različita poduzeća koja nude istu uslugu i korisnik odabire poduzeće koje mu najviše odgovara za izvršavanje te usluge,

- **poduzeće-država** (eng. business – to – government) – varijanta poduzeće – poduzeće modela, vlada koristi ovakav model kako bi razmijenila informacije s raznim poduzećima. Ove stranice služe kao sredstvo poduzećima za podnošenje prijava za državne poslove,

- **država-poduzeće** (eng. government – to – business) – vlade koriste ovaj model kako bi prišle poduzećima. Koriste se za organiziranje vladinih aukcija i poslovnih ponuda,

- **država-stanovnik** (eng. government – to – citizen) – vlade koriste ovaj model kako bi prišle građanima. Koriste se za organiziranje aukcija za državna vozila, te služe za izdavanje različitih dokumenata građanima.

Sve E-trgovine koriste sustav elektroničkog plaćanja koji je smanjuje količinu papirologije, cijenu transakcije i rada potrebnog za obavljanje kupovine. Elektroničko plaćanje omogućuje korisniku brži i lakši način plaćanja i omogućuje poduzećima širenje njihovih tržišta. Neki od modela elektroničkog plaćanja su: kreditne kartice, debitne kartice, e-novac, elektronički prijenos sredstava, mobilno plaćanje i elektronički novčanik. Najpopularniji način elektroničkog plaćanja je plaćanje kreditnim i debitnim karticama. Kreditne i debitne kartice su male plastične kartice s jedinstevnim brojem koji je povezan sa bankovnim računom korisnika. Razlika je u tome što pri plaćanju kreditnom karticom banka plati traženi iznos, a korisnik ima neko vrijeme da vrati taj novac banci, dok kod plaćanja debitnom karticom korisnik mora u trenutku plaćanja na računu imati dovoljan iznos novca za obavljanje transakcije. E-novac je oblik novca koji nije dostupan u fizičkom obliku već samo u digitalnom obliku. Transakcije e-novca odnose se na situacije kada se plaćanje odvija preko interneta i iznos se prebacuje s jednog financijskog tijela na drugo bez posrednika. Elektronički prijenos sredstava je model elektronskog plaćanja u kojem se novac prebacuje s jednog bankovnog računa na drugi bankovni račun. Elektronički novčanik je oblik računa na stranici e-trgovine u koji korisnik može uplatiti e-novac i pomoću njega obavaljati kupovinu. Mobilno plaćanje je oblik elektroničkog plaćanja koje se vrši pomoću mobilnog uređaja.

U ovom radu koristiti će se model platnih kartica, stoga je bitno pobliže se upoznati sa tim modelom.

3 Model plaćanja platnim karticama

Sustav plaćanja (eng. payment system) platnim karticama sastoji se od pet ključnih teorijskih uloga, a to su:

- **vlasnik kartice** (eng. cardholder) – osoba koja posjeduje platnu karticu i koristi tu karticu za kupovinu,
- **trgovac** (eng. merchant) – prodavač proizvoda kojeg vlasnik kartice želi kupiti, donosi odluke o tome koju marku platne kartice želi prihvatiti. Odgovoran je za osiguravanja podataka o vlasniku kartice,
- **banka izdavatelj kartice** (eng. card issuer bank) – banka koja proizvodi i distribuira platne kartice korisnicima. Izdaje podatke o vlasnicima kartica,
- **banka stjecatelja** (eng. acquirer bank) – banka koja obavlja proces plaćanja platnom karticom u ime trgovca, to jest prima iznos novca koji vlasnik kartice koristi za kupovinu proizvoda,
- **brandovi platnih kartica** (eng. card brands) - igraju centralnu ulogu u plaćanju kreditnim karticama. Olakšavaju proces plaćanja i povezivanje između banke izdavatelja kartice i banke stjecatelja. Brandovi platnih kartica razvili su i PCI DSS [1] kako bi održali sigurnosne standarde kojima trgovac osigurava podatke o vlasniku kartice. Neki od najpoznatijih brandova su: Visa, MasterCard, American Express, Diners Club i tako dalje.

Uz tih pet uloga u praksi se pojavljuju i **obrađivač plaćanja** (eng. payment processor), **usmjerivač plaćanja** (eng. payment gateway), **programska podrška** (eng. software) i **sklopovlje** (eng. hardware).

3.1 Obrađivač plaćanja

Obrađivač plaćanja (eng. payment processor) je tvrtka koja obrađuje transakcije platnim karticama između trgovca i kupca. Dijele se na dva tipa obrađivača: front-end i back-end obrađivač. Front-end obrađivač prikuplja detalje vezane za kupca i njegovu karticu i šalje ih

[1] Standard za sigurnost podataka industrije platnih kartica (eng. Payment Card Industry Data Security Standard

banci koja je izadala karticu na verifikaciju. Zatim provodi niz mjera protiv prijevare kojima se dodatno želi pojačati sigurnost transakcije. Nakon što je potvrđeno da je transakcija valjana aktivira se back-end obrađivač. On prihvaća transakciju i šalje novac od banke kupca do banke stjecatelja. Nakon što obrađivač plaćanja primi informaciju da je kartica prihvaćena i da je obavljen prijenos novca poziva se *usmjerivač plaćanja* (eng. payment gateway) koji tu informaciju šalje trgovcu kako bi se transakcija privela kraju. U slučaju da kartica nije prihvaćena trgovac odbija transakciju.

Obrađivači plaćanja pridržavaju se određenih standarda i regulacija koje nameću tvrtke kreditnih kartica. Ti standardi uključuju pravila vezana za prijevare, povrat novaca i krađu identiteta.

3.2 Usmjerivač plaćanja

Usmjerivač plaćanja (eng. payment gateway) je servis koji služi za obradu online plaćanja. Funkcionira poput sigurnog mosta koji služi za prijenos informacija vezanih za transakciju između banke izdavatelja kartice i banke stjecatelja. Prilikom online kupovine usmjerivač plaćanja sudjeluje u sljedećim funkcijama:

- **autorizacija** (eng. authorising) – provjera i potvrda valjanosti platne kartice kupca,
- **čišćenje**(eng. clearing) – slanje novca banci stjecatelja,
- **izvještavanje** (eng. reporting) – praćenje svih transakcija.

Neki usmjerivači plaćanja posjeduju i dodatne alate koji korisnicima omogućuju izračunavanje poreza i troškova dostave. Također sadrže i razne mjere protiv prijevara i krađa informacija vezanih za platne kartice. Implementacija usmjerivača plaćanja nije obavezna za e-trgovine, ali je najrašireniji način povezivanja banaka. Samo najveće kompanije mogu se izravno povezati s bankama bez korištenja usmjerivača plaćanja.

Upoznajmo se sada s nekim metodama integriranja usmjerivača plaćanja.

3.2.1 Metode integriranja usmjerivača plaćanja

U teoriji postoje dva načina implementiranja usmjerivača plaćanja. Prvi način odvija se na serveru usmjerivača plaćanja. Tijekom procesa provjere (eng. checkout) na stranici

e-trgovine kupac popunjava određenu formu unutar koje unosi svoje osobne podatke. Nakon toga preusmjerava se na stranicu usmjerivača plaćanja kako bi unio podatke o platnoj kartici. Kada popuni te podatke vraća se nazad na stranicu e-trgovine kako bi završio kupovinu. Pošto se unos osjetljivih informacija o platnoj kartici vrši na sigurnom serveru usmjerivača plaćanja sama e-trgovina ne mora imati SSL[2] certifikat. Ovaj način implementacije pruža dobru zaštitu podataka jer se povjerljivi podaci upisuju na serveru usmjerivača plaćanja i stoga sama e-trgovina nema nikakve odgovornosti prema tim podacima. Jedna od mana ovog načina jest nemogućnost dizajnirana obrasca za plaćanje jer se on nalazi na stranici usmjerivača plaćanja i također korisničko iskustvo je narušeno zbog preusmjeravanja korisnika na drugu stranicu. Drugi način implementacije je pomoću *sučelja za programiranje aplikacija* (eng. application programming interface) ili skraćeno API-a kojeg pružaju tvrtke poput PayPal-a. U ovoj implementaciji kupac je na stranici e-trgovine tijekom cijelog proces kupovine. Nakon što unese svoje podatke i podatke o platnoj kartici oni se šalju na server usmjerivača plaćanja bez preusmjeravanja korisnika na stranicu usmjerivača plaćanja. Na ovaj način korisnik ima percepciju da je sve obavljeno na stranici e-trgovine. Kako bi se zaštitili kupčevi podaci e-trgovina mora imati SSL certifikat i također bilo bi dobro da podatke enkriptira prilikom slanja usmjerivaču plaćanja. Prednost ovog načina je da kupac nikad ne mora napustiti stranicu e-trgovine i na taj način je poboljšano korisničko iskustvo. Veliki nedostatak ovog načina implementiranja jest što je e-trgovina odgovorna za sigurnost podataka vezanih za kreditnu karticu kupca. Također prije implementacije potrebno je položiti određene certifikate o sigurnosti koje imaju banke. Zbog toga je ovaj način vrlo skup i tehnički jako zahtjevan. U praksi se koristi kombinacija ova dva načina implementiranja.

Pogledajmo sada neke od metoda implementiranja.

Jednostavna metoda plaćanja (eng. simple checkout method)

U ovoj metodi usmjerivač plaćanja integriran je kao vanjski servis (eng. external service). Kada kupac klikne na „Kupi“ preusmjerava se na stranicu *pružatelja usluga usmjerivača plaćanja* (eng. payment gateway service provider) kako bi obavio sigurnu transakciju. Ovo je najjednostavnija metoda za implementiranje. Prednosti ove metode su što se ne mora plaćati održavanje usmjerivača plaćanja i ne snosi se odgovornost za sigurnost podataka

[2] *Protokol za prijenos kodiranih podataka* (eng. secure socket layer) je sigurnosni protokol koji osigurava prijenos osobnih i povjerljivih podataka putem web-a.

kupca. Također stranica ne mora biti PCI DSS suglasna. Nedostatak je to što korisnik mora napuštati našu stranicu. Također sklona je *čovjek-u-sredini*[3] (eng. man-in-the-middle) napadima. Ova metoda savršeno je rješenje za mala poduzeća.

Metoda direktnog slanja (eng. dircet post method)

U ovoj metodi forma koja prima informacije o kupcu i detalje o transakciji i kreditnoj kartici nalazi se na serveru e-trgovine. Nakon što korisnik unese sve podatke oni se šalju usmjerivaču plaćanja, gdje se obavlja transakcija. Podaci se nikad ne spremaju u bazi podataka e-trgovine. Metoda direktnog slanja pruža trgovcu slobodu oko dizajna forme za plaćanje jer se ona nalazi na njegovom serveru. Isto kao i jednostavna metoda ni ova metoda ne mora bit PCI DSS suglasna. Nedostatak ove metode je što nije sigurna i može lako biti komprimirana.

Metoda serverske integracija (eng. server integration method)

Ova metoda omogućuje trgovcu da se transakcija prividno obavlja na njegovoj stranici dok u pozadini usmjerivač plaćanja obavlja transakciju. Sve forme za unos podataka nalaze se na stranici e-trgovine. Svi podaci vezani za korisnika i kreditne kartice spremaju se na server e-trgovine. Usmjerivač plaćanja obavlja sve korake u procesu sigurne transakcije kao što su prikupljanje podataka o transakciji i odgovor kupcu, ali kupac to ne zna. Na ovaj način trgovac može urediti sve forme i račun po svojoj želji. Za ovu metodu e-trgovina mora biti PCI DSS suglasna i mora imati SSL certifikat i također mora poboljšati sigurnosne standarde jer će upravljati podacima vezanim za kupce. Ovu metodu korisne poduzeća srednje veličine koja imaju razvijen svoj brand.

Napredna metoda integracije (eng. advanced integration method)

Ova metoda omogućuje trgovcu potpunu kontrolu nad procesom realiziranja transakcije. Sve forme vezane za transakciju nalaze se na stranici e-trgovine. E-trgovina prati korisnika od njegovog dolaska na stranicu do njegovog odlaska sa stranice. Podaci vezani za kupca i platne kartice spremaju se u bazu podataka e-trgovine. Jedini posao usmjerivača plaćanja kod ove metode je prijenos novca, sve ostalo vrši sama e-trgovina. Za ovu metodu potreban je SSL certifikat i PCI DSS suglasnost i koriste ju sam najveće tvrtke na svijetu.

[3]Zlonamjerni korisnik može ometati komunikaciju između dva korisnika

3.2.2 Proces online kupovine

Upoznajmo se s procesom kupovine u kojem sudjeluje usmjerivač plaćanja (vidi slika 1). Sastoji se od 3 koraka:

1. **prikupljanje podataka** (eng. collection):

   Kada kupac dođe na stranicu e-trgovine i odluči kupiti određeni proizvod preusmjerava se na stranicu za plaćanje gdje unosi podatke vezane za platnu karticu. Ukoliko e-trgovina zadovoljava potrebne sigurnosne standarde stranica za plaćanje može se generirati na stranci e-trgovine, ukoliko ne zadovoljava sigurnosne standarde stranica za plaćanje generira se na stranici usmjerivača plaćanja. U oba slučaja informacije o platnoj kartici i transakciji spremaju se na sigurnim serverima. Nakon prikupljanja potrebnih informacija usmjerivač plaćanja šalje te podatke putem SSL veze obrađivaču plaćanja.

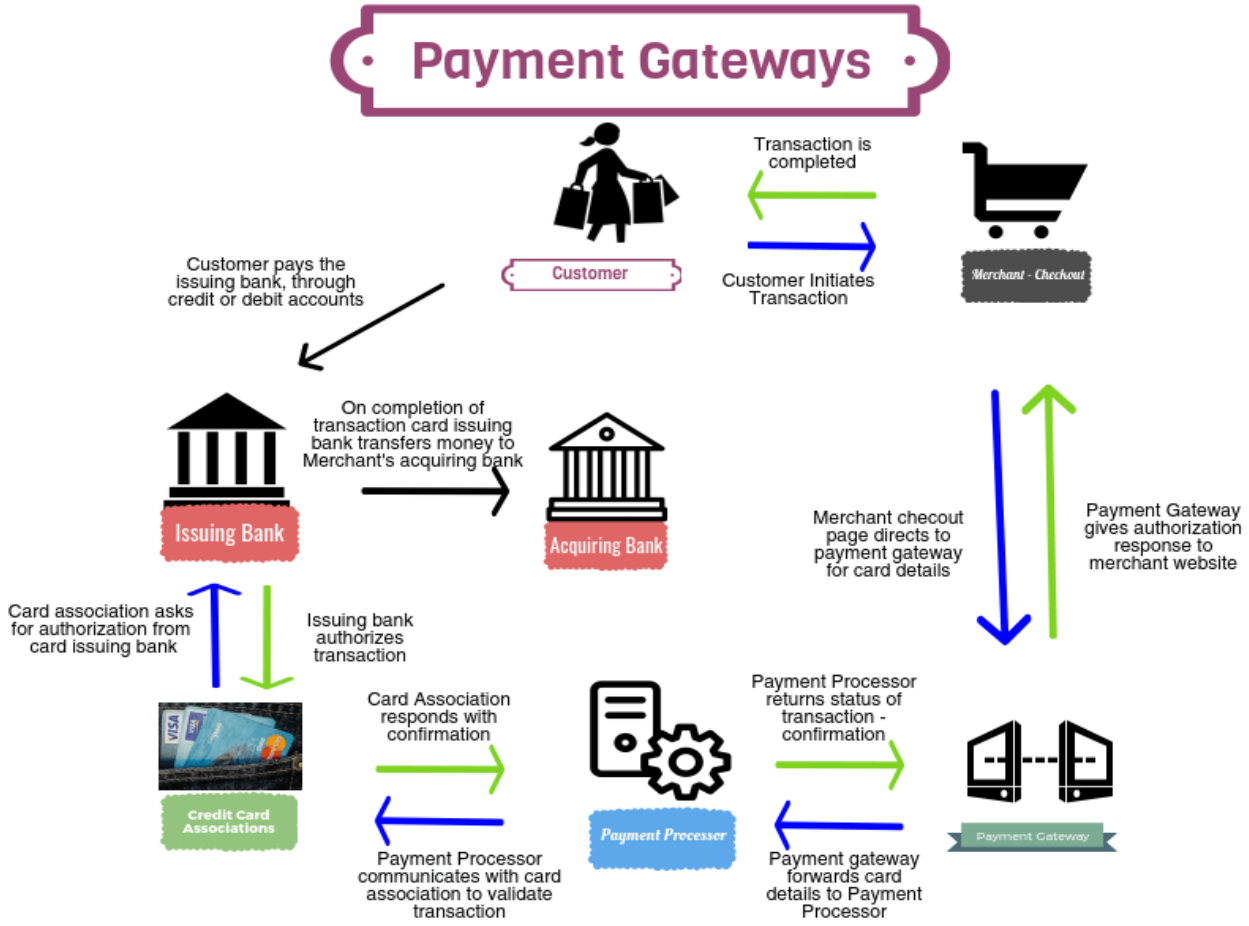
2. **autentifikacija i autorizacija** (eng. authentication and authorization):

   Nakon što primi podatke od strane usmjerivača plaćanja obrađivač šalje banci stjecatelja te podatke i traži autorizaciju i autentifikaciju podataka. Zatim banka stjecatelja kontaktira banku izdavatelja kartice kako bi potvrdila da kupac ima dovoljan iznos novca za obavljanje transakcije i kako bi potvrdila podatke vezane za kupca. Kada prođe proces autorizacije kartice banka izdavatelj šalje odgovor banci stjecatelja koja onda taj odgovor prosljeđuje nazad obrađivaču plaćanja. Obrađivač preusmjerava podatke na usmjerivač plaćanja koji ih obrađuje. Usmjerivač plaćanja priprema svoj odgovor za slanje e-trgovini ne samo na osnovu toga je li autorizacija prošla nego provodi i druge testove kojima utvrđuje je li kreditna kartica ukraden i radi li se o prijevari. Nakon tih testova usmjerivač plaćanja sprema detalje o transakciji i šalje odgovor e-trgovini. E-trgovina prikazuje poruku o prihvaćanju ili odbijanju narudžbe ovisno o odgovoru primljenom od usmjerivača plaćanja i nakon toga sprema narudžbu u svoju bazu podataka.

3. **čišćenje transakcije** (eng. clearing transaction):

   Nakon procesa autoriziranja preostaje izvršiti transfer novca s računa kupca na račun trgovca. To nazivamo proces nagodbe (eng. settlement). Trgovac šalje transakciju svojoj banci. Zatim ta banka stvara zahtjev za nagodbu koji šalje banci izdavatelja

kartice. Banka izdavatelj kartice skida iznos potreban za naplatu transakcije s računa kupca i šalje ga banci stjecatelja koja taj iznos stavlja na račun trgovca i time završava kupovina.



Slika 1: Proces online kupovine

3.2.3 PCI DSS standard

Standard sigurnosti podataka industrije platnih kartica (eng. payment card industry data security standard) je globalni standard kojim se regulira kartična sigurnost i sigurnost korisnika prilikom online plaćanja. PCI DSS standard izrađen je 2004. godine, a u njegovoj izradi sujdelovalo je pet tvrtki: Visa, MasterCard, Discover, America Express i JCB. Nastao je s namjerom da se stvori dodatna razina zaštite izdavateljima kartica osiguravajući da tvrtke zadovoljavaju minimalnu razinu sigurnosti kada obavljaju spremanje, obradu i prijenos podataka s

kartice. Zahtjevi PCI DSS standarda podijeljeni su u šest logičkih poglavlja.

Izgradnja i održavanje sigurne mrežne infrastrukture

- Zahtjev 1: Ugraditi i održavati vatrozid podešen tako da zaštiti kartične podatke.
- Zahtjev 2: Ne koristiti lozinke i druge sigurnosne parametre dobivene od prodavača sklopovlja i programa.

Zaštita kartičnih (korisničkih) podataka

- Zahtjev 1: Zaštiti spremljene kartične podatke.
- Zahtjev 2: Šifrirati kartične podatke za vrijeme prijenosa po otvorenim, javnim mrežama.

Održavanje programa za upravljanje ranjivostima

- Zahtjev 1: Koristiti i redovito ažurirati anti-virusne programske pakete.
- Zahtjev 2: Razvijati i održavati sigurne sustave i aplikacije.

Implementacija jakih provjera sustava

- Zahtjev 1: Ograničiti pristup kartičnim podacima modeliranjem poslovnog procesa.
- Zahtjev 2: Dodijeliti jedinstveni ID svakoj osobi koja pristupa računalu.
- Zahtjev 3: Ograničiti fizički pristup kartičnim podacima.

Redoviti nadzor i ispitivanje mrežne infrastrukture

- Zahtjev 1: Pratiti i provjeravati svaki pristup mrežnim resursima i kartičnim podacima.
- Zahtjev 2: Redovito provjeravati sigurnost sustava i procesa.

Održavanje sigurnosne politike

- Zahtjev 1: Održavati pravilnik koji se odnosi na informacijsku sigurnost.

(PCI Security Standard Council 2010)

4 Izrada aplikacije

Praktični dio diplomskog rada je implementacija elektroničkog plaćanja pomoću PayPal Express Checkout servisa i testiranje istog na PayPal sandbox okruženju. Aplikacija ima osnovne funkcije e-trgovine. Administrator e-trgovine može dodavati, uređivati i brisati proizvode. Gosti koji dolaze kao i registrirani korisnici mogu dodati željene proizovde u košaricu te kupiti iste putem PayPal-a. Aplikacija je izrađena je koristeći razvoji alata Visual Studio i ASP.NET tehnologiju.

4.1 Visual Studio 2017 i ASP.NET

Visual Studio je *integrirano razvojno okruženje* (eng. Integrated development environment) koje se koristi se za izradu desktop, web i mobilnih aplikacija. U Visual Studio ugrađen je Microsoft-ov .NET okvir (eng.framework). Uključuje *uređivač kôda* (eng. code editor) koji podržava IntelliSense[4] te ugrađeni *program za ispravljanje grešaka* (eng. debugger) koji radi *na strojnoj razini* (eng. machine-level debugger) i *na razini kôda* (eng. source-level debbuger). Visual Studio sadrži još mnoštvo korisnih alata kao što su dizajner klasa, dizajner sheme baze podataka i NuGet upravitelj paketa. Podržava velik broj programskih jezika među kojima su C, C++, C#, JavaScript, HTML i tako dalje. Za izradu aplikacije korišten je Visual Studio 2017 Community Edition koji je besplatan, a kôd je pisan u C# programskom jeziku.

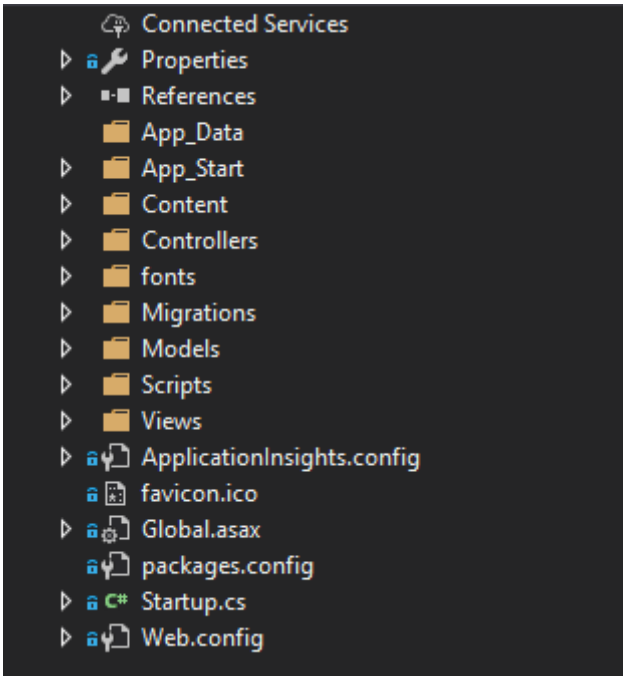
ASP.NET je *web okvir* (eng. framework) koji služi za izradu web aplikacija. Razvijen je od strane Microsoft-a. Izrađen je na Common Language Runtime (CLR) tehnologiji što omogućava razvojnim programerima da koriste bilo koji od podržanih .NET programskih jezika. Za izradu aplikacije korištena je ASP.NET *Model-Pogled-Kontroler* (eng model-view-controller) arhitektura koja dijeli aplikaciju na tri glavne komponente:

- **Modeli** – objekti koji implementiraju logiku za domenu aplikacije, dohvaćaju podatke iz baze podataka i spremaju podatke u bazu podataka,
- **Pogledi** – komponente aplikacije koje služe za prikaz korisničkog sučelja (eng. user interface) aplikacije,

[4]Generalni naziv za razna pomagala u pisanju kôda (automatsko dovršavanje kôda, informacije o parametrima i td.)

- **Kontroleri** – komponente koje reagiraju na korisničke zahtjeve, rade s modelima i odabiru koji će se pogled prikazati.

Prilikom pokretanja novog projekta u Visual Studio-u dobivamo standardnu strukturu jedne ASP.NET MVC aplikacije. (vidi slika 2)



Slika 2: Standardna struktura ASP.NET aplikacije

Unutar mape **App_Start** nalaze se datoteke koje se pokreću prilikom pokretanja aplikacije, kao što su **RouteConfig.cs** u kojoj su definirane rute (eng. routes) i **BundleConfig.cs** u kojoj su definirani sve skripte koje se koriste u aplikaciji poput bootstrap-a i javaScript-a. Kao što i sama imena govore unutar mapa **Models**, **Controllers**, **Views** spremaju se redom modeli (klase i objekte), kontroleri i pogledi. Unutar mape **Models** nalazi se datoteka **IdentityModels.cs** koja sadrži klasu **ApplicationDbContext** koja se koristiti za kreiranje migracija prema bazi podataka. Datotek **Web.config** poslužit će prilikom definiranja konfiguracije za povezivanje s PayPal-om.

4.2 WebShop aplikacija

Kako bih implementacija PayPal Exprees Checkout servis bila moguća potrebno je napravati prototip e-trgovine. Aplikacija treba imati osnovne funkcije e-trgovine kao što su dodavanje i uređivanje proizvoda i naravno mora imati implementirani proces kupovine.

Kako se cijela aplikacija fokusira na prozivodima bitno je najprije kreirati klasu **Product.cs** (vidi kod 1) koja služi za instanciranje proizvoda i osnovni je model same aplikacije.

```
public class Product
    {
        public int Id { get; set; }

        [Required]
        [StringLength(255)]
        public string Name { get; set; }

        [Required]
        public decimal Price { get; set; }
    }
```

Kod 1: Klasa **Product**

Klasa **Product** sastoji se od tri atributa **Id**, **Name**, **Price** u koje se redom spremaju identifikator, naziv i cijena proizvoda. Unutar uglatih zagrada nalaze se anotacije (eng. annotations) kojima se dodatno opisuju ti atributi. Na primjer iznad atributa **Name** nalaze se antonacije kojima se naglašva da se prilikom definiranja objekta proizvod mora navesti atribut **Name** i on može biti maksimalne duljine od 255 znakova. Primijetimo da iznad atributa **Id** nemamo nikakvih anotacija, razlog tome je što ASP.NET prepoznaje atribut **Id** kao identifikator i sam mu dodaje vrijednost prilikom unošenja u bazu podataka. Preostaje pomoću migracije (eng. migration) kreirati tablicu u bazi podataka na osnovu klase **Product**. Unutar **IdentityModels.cs** klasi **ApplicationDbContext** se inicijalizira novi **DbSet** objekt baziran na Product klasi (vidi Kod 2).

```
public DbSet<Product> Products { get; set; }
```

Kod 2: Inicijalizacija novog DbSet (tablice baze podataka) objekta

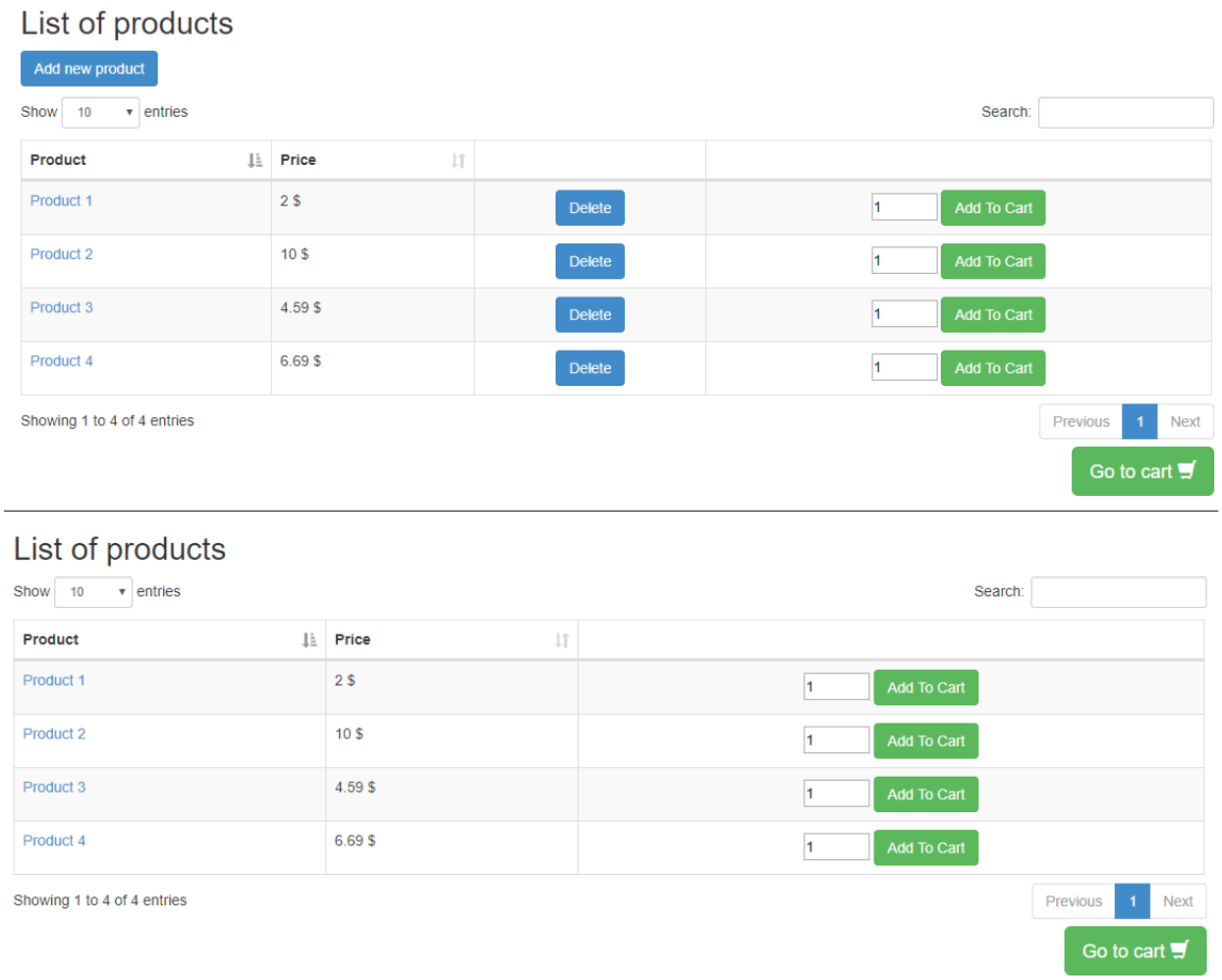
Na taj način govorimo ASP.NET-u da želimo definirati novu tablicu **Products** unutar baze podatka koju stvaramo iz klase **Product**. Koristeći konzolu kreira se nova migracija naredbom **add-migration „naziv migracije“**. Unutar migracije ASP.NET automatski kreira tablicu pazeći pritom na anotacije i tipove varijabli. Na primjer ako atribut ima anotaciju **required** onda prilikom kreiranja tablice taj stupac u tablici ne može poprimiti *nul vrijednost* (eng. null).(vidi kod 3).

```
CreateTable(
    "dbo.Products",
    c => new
        {
            Id = c.Int(nullable: false, identity: true),
            Name = c.String(nullable: false, maxLength: 255),
            Price = c.Decimal(nullable: false, precision: 18, scale: 2),
        })
    .PrimaryKey(t => t.Id);
```

Kod 3: Kreiranje tablice unutar migracije

Nakon kreiranja tablice potrebno je istu ubaciti u bazu podataka koristeći nardebu **update-database**. Definiranjem klase proizvod i tablice proizvoda unutar baze podataka definiran je model aplikacije. Preostaje definirati pogled i kontroler koji će prikazati tablicu proizvoda korisniku. Aplikacija ima funkcije poput brisanja i uređivanja proizvoda koje ne bi trebale biti dostupne običnom korisniku već samo administratoru stoga treba napraviti različite poglede koji će se prikazivati ovisno o ovlastima pojedinog korisnika.

U tu svrhu definiraju se dva pogleda, jedan za administratora i jedan za goste odnosno kupce. Oba pogleda prikazuju tablicu proizvoda koja ima ugrađene funkcije sortiranja po stupcima, pretraživanja i obilježavanja stranica. Administrator će imati gumb za dodavanje novog proizvoda, gumb za brisanje proizvoda i također će klikom na naziv proizvoda biti preusmjeren na stranicu za uređivanje istog, dok će korisnici imati samo opciju dodavanja proizvoda u košaricu. (vidi slika 3)



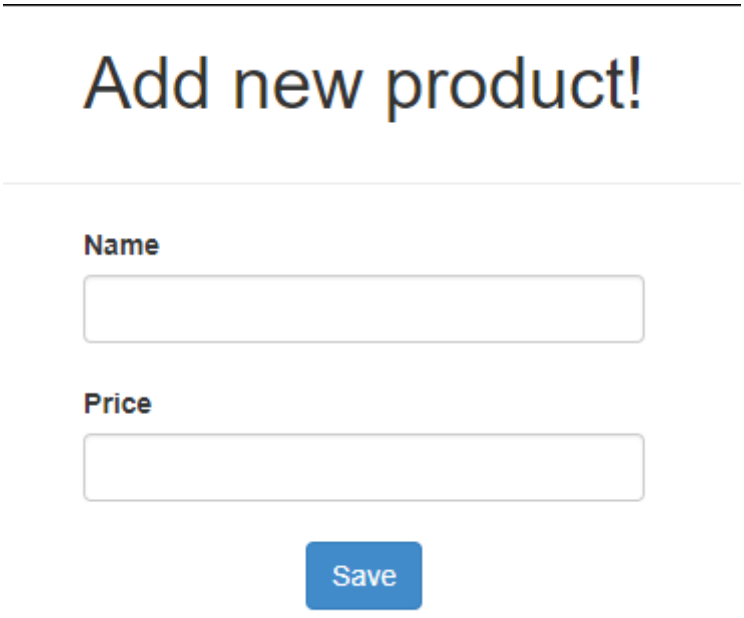
Slika 3: Pogled za admina i pogled za kupce

Ovaj način autoriziranja pogleda ostavlja velike sigurnosne propuste. Naime, korisnik koji nema administratorske ovlasti još uvijek može pristupiti formi za dodavanje novog proizvoda kao i formi za uređivanje proizvoda pomoću *usklađenog lokatora resursa* (eng. uniform resource locator) ili skraćeno URL-a. Ako korisnik poznaje rute koje koristimo kako bi pristupili određenom pogledu može ih jednostavno upisati u URL i tako pristupiti svim pogledima. Ovaj problem se rješava tako da se unutar kontrolera iznad svake akcije (eng. action) doda anotacija **Authorize** koja definira uloge (eng. roles) koje mogu pristupiti određenoj akciji. (vidi kod 4)

```
[Authorize(Roles = RoleName.Admin)]
        public ActionResult Update(Product product)
        {
          ...
        }
```

Kod 4: Dodavanje anotacije **Authorize** (**RoleName** je klasa unutar koje definiramo uloge, a **CanManageProducts** je atributa klase koji smo definirali)

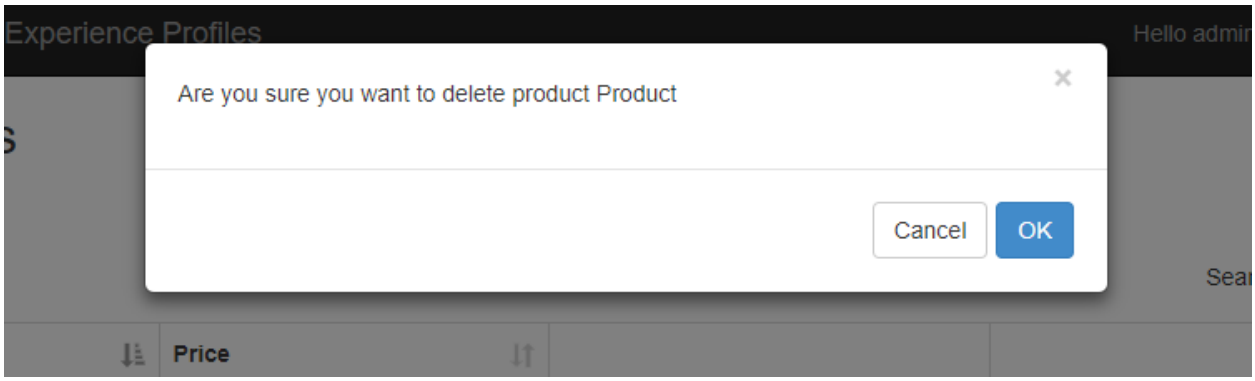
Sada je potrebno definirati poglede za dodavanje i uređivanje proizvoda. U osnovi ta dva pogleda dijele isti dizajn (vidi slika 4). Kada se želi dodati novi proizvod koristi se gumb **Add new product** (vidi slika 3) koji vodi do forme za dodavanje potrebnih informacija o proizvodu. Iz definicije klase **Product** jasno je da forma treba sadržavati dva okvira za unos jer klasa sadrži dva atributa **Name** i **Price**. Treba voditi računa o validaciji koja će u slučaju greške biti ispisana ispod polja u kojem je napravljena pogreška. Kada administrator želi urediti proizvod klikom na ime proizvoda u tablici proizvoda dolazi do forme za uređivanje koja izgleda identično kao forma za dodavanje, jedina razlika je u tome što je forma za uređivanje unaprijed popunjena s informacijama proizvoda koji se želi urediti.



Slika 4: Forma za dodavnje novog proizvoda

Za brisanje određenog proizvoda koristi se **Delete** gumb u tablici pored proizvoda koji se želi obrisati (vidi slika 3). Klikom na taj gumb ne otvara se novi pogled nego iskače

dijaloški okvir (eng. dialog box) koji upituje korisnika je li siguran da želi obrisati određeni proizvod (vidi slika 5). Dijaloški okvir dobiven je korištenjem bootbox-a[5].



Slika 5: Primjer brisana proizvoda iz tablice proizvoda

Kada kupac želi dodati proizvod u košaricu čini to pritiskom na gumb **Add to cart** pored prozivda koji želi dodati. Nakon što doda sve željene proizvode može posjetiti košaricu klikom na **Go to cart** gumb (vidi slika 3)koji ga vodi na pogled koji prikazuje detalje o njegovoj košarici (vidi 6).

Slika 6: Prikaz košarice

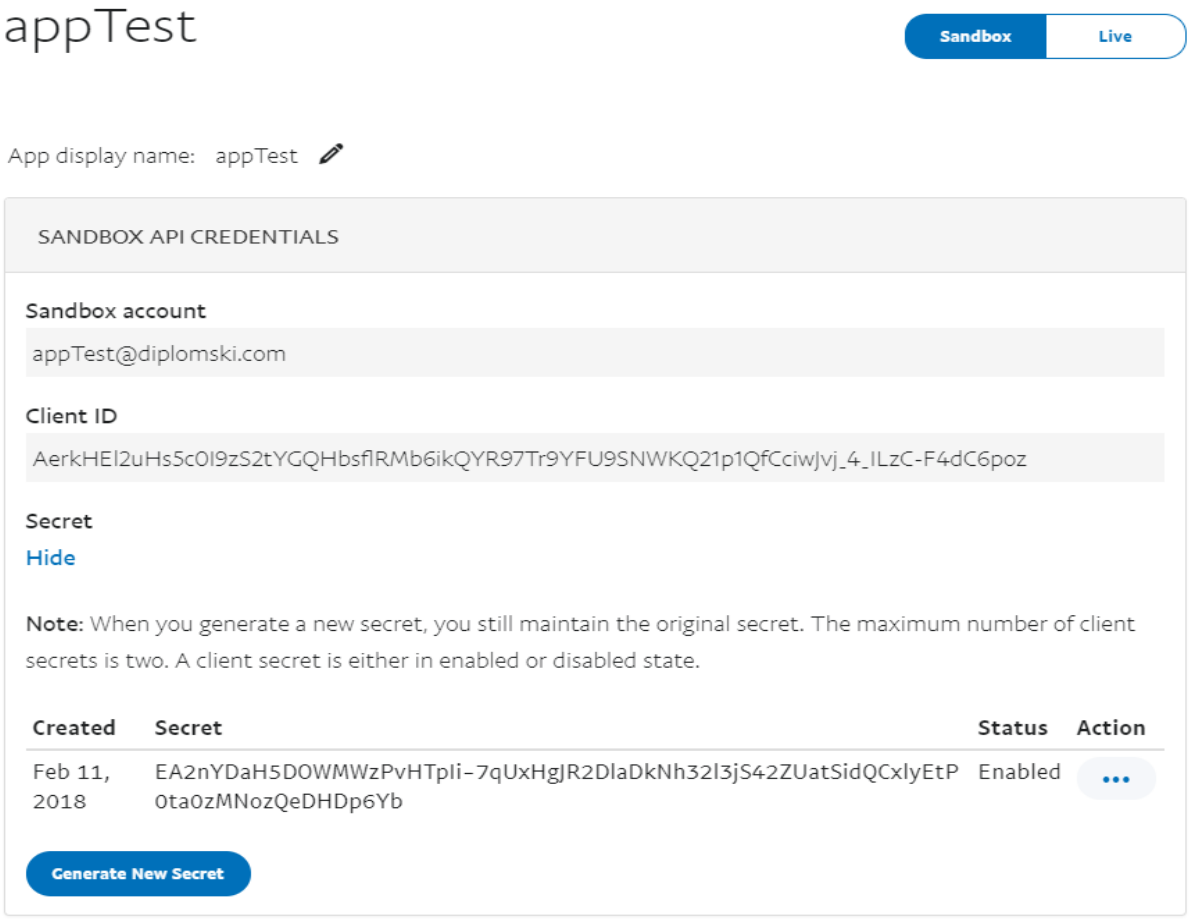
Košarica pokazuje informacije o nazivu, cijenu i kvantiteti proizvoda. Klikom na **Remove** kvantiteta se smanjuje za jedan, a u koliko je kvantiteta jednaka jedan onda se klikom na **Remove** briše proizvod iz košarice. U zadnjem retku tablice prikazana je ukupna cijena košarice i gumbi za nastavak kupovine i završetak kupovine.

[5]Bootbox je JavaScript biblioteka koja nam omogućuje stvaranje dijaloških okvira

Sada su definirane sve funkcionalnosti aplikacije vezane za upravljanje tablicom proizvoda i kreiranje košarice te može početi implementacija PayPal Express Checkout servisa. No, prije toga preostaje osposobiti PayPal račun kako bi se aplikacija mogla povezati s PayPal-om i kako bi se kasnije mogla obaviti test kupovina putem PayPal sandbox okruženja.

4.3 Osposobljavanje PayPal Developer računa

Prije implementacije PayPal Express Checkout servis u web aplikaciju potrebno je kreirati PayPal poslovni račun. Nakon kreiranja PayPal računa treba se prijaviti na PayPal Developer gdje na *kontrolnoj ploči* (eng. dashboard) postoje opcije za kreiranja PayPal aplikacije i sanbox računa koji će kasnije poslužiti u testiranju same web aplikacije. Sam proces kreiranja sandbox računa je prilično jednostavan, popunjavanjem dane forme kreira se test račun s željenim iznosom novca. Još je preostalo kreirati aplikaciju na PayPal sandbox okruženju. Odabirom opcije **"My apps and Credentials"** na kontrolnoj te odlaskom na **"REST API apps"** dolazi se do gumba za kreiranje PayPal test aplikacije. Kreiranjem aplikacije na PayPal servisu dobiju se dva ključa „Client ID“ i „Secret“ koji će kasnije služiti za povezivanje web aplikacije s PayPal-om. (vidi slika 7).



Slika 7: Primjer PayPal sandbox aplikacije i njenih ključeva

Nakon kreiranja PayPal aplikacije potrebno je osposobiti podršku za PayPal unutar Visual Studio-a. Koristeći NuGet *upravitelj paketima* (eng. package manager) instalira se PayPal-ov *set razvojnih alata* (eng. software development kit) ili skraćeno SDK . Preostaje dodati konfiguraciju za PayPal unutar **web.config** dokumenta kako bi se mogla uspostaviti komunikaciji između aplikacije i PayPal servisa (vidi kod 5).

```
<paypal>
    <settings>
      <add name="mode" value="sandbox" />
      <add name="clientId" value="Aa2_Udlhebpyw5EGODj..." />
      <add name="clientSecret" value="EO6uqfmr1Mj7E1SN1..." />
    </settings>
</paypal>
```

Kod 5: Konfiguracija za PayPal unutar web.config dokumenta

Unutar konfiguracije nalaze se tri čvora (eng. node). Čvor s imenom (eng. name) „način“ (eng. mode) koji nam govori da se radi o sandbox aplikaciji, a ne o aktivnoj (eng. live) aplikaciji. Zatim slijede dva čvora s imenima **clientId** i **clientSecret** koji služe za spremanje ključeva dobivenih u PayPal aplikaciji. Sada je ostvarena komunikacija između aplikacije i PayPal servisa i može se započeti s implementacijom PayPal Express Checkout servisa.

4.4 Integracija PayPal Express Checkout servisa

PayPal Express Checkout je PayPal-ov servis koji korisniku pruža pojednostavljeno kupovanje putem e-trgovine. Kupac tijekom kupovine boravi na stranici e-trgovine, odabire koje proizvode i usluge želi kupiti i nakon toga je preusmjeren na stranicu PayPal-a gdje završava proces plaćanja. Nakon toga kupac je ponovno preusmjeren na stranicu e-trgovine gdje se izvršava prijenos novca i time kupovina završava. Na ovaj način kupac u nijednom trenutku ne mora dijeliti osjetljive informacije vezane za kreditne kartice i bankovne račune na stranici e-trgovine što čini kupovinu mnogo sigurnijom.

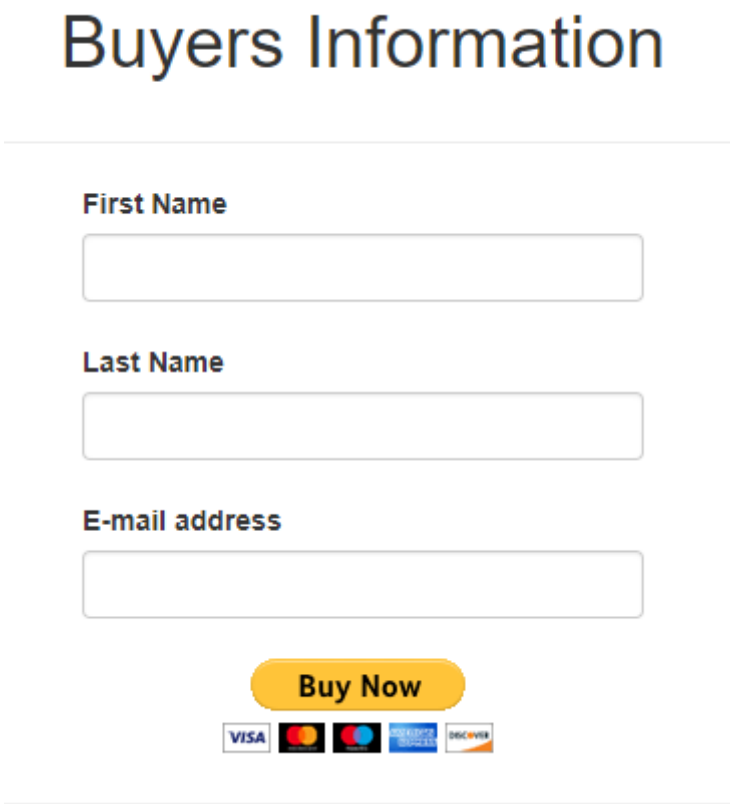
Za implementiranje Express Checkout servis treba najprije kreirati novu klasu koja će se koristiti za spremanje informacija vezanih za kupovinu određenog proizvoda. Ta klasa zvati će se **OrderInfo** te će sadržavati 3 atributa vezana za kupca (**FirstName**, **LastName**, **Email**) i referencu za PayPal koja služi za spremanje PayPal ID-a koji se dobiva od strane PayPal-a za svaku kupovinu. Pomoću tog ID-a kasnije će se pristpuati dodatnim informacijama vezanim za transakciju. (vidi kod 6)

```
1  public class OrderInfo
2      {
3          public int Id { get; set; }
4
5          [Required]
6          [Display(Name = "First Name")]
7          public string FirstName { get; set; }
8
9          [Required]
10         [Display(Name = "Last Name")]
11         public string LastName { get; set; }
12
13         [Required]
```

```
14          [Display(Name = "E-mail address")]
15          public string Email { get; set; }
16
17          public string PayPalReference { get; set; }
18      }
```

Kod 6: Klasa **ProductOrderInfo**

Kada kupac završi s popunjavanjem košarice i klikne na **Checkout** (vidi slika 6) preusmjeren je na stranicu za prikupljanje informacija potrebnih za nastavak kupovine. Stranica se sastoji od forme koja od kupca zahtjeva sljedeće informaijce: ime, prezime i e-mail adresu. (vidi slika 8)



Slika 8: Forma za kupovinu proizvoda

Nakon što kupac unese svoje informacije i klikne na **Buy Now** poziva se **PaymentAction** (vidi 7) funkcija unutar **ProductPaymentController** kontrolera.

```
[HttpPost]
public ActionResult PaymentAction(OrderInfo model)
    {
        if (ModelState.IsValid)
        {
            var cart = Session["Cart"] as Cart;
            // Create a ProductOrderInfo object

            var orderInfo = new OrderInfo()
            {
                FirstName = model.FirstName,
                LastName = model.LastName,
                Email = model.Email
            };

            var totalPrice = cart.ProductsInCart.Sum(x => x.Product.Price*x.
    Quantity);

            _context.ProductOrderInfoes.Add(orderInfo);
            _context.SaveChanges();

            var apiContext = GetApiContext();

            // Create a new payment object
            var payment = new Payment
            {
                experience_profile_id = "XP-9LYR-B3CK-PD8V-Z58W",
                intent = "sale",
                payer = new Payer
                {
                    payment_method = "paypal"
                },
                transactions = new List<Transaction>
                {
                    new Transaction
                    {
```

```
                        description = "Thank you for your purchase!",
                        amount = new Amount
                        {
                            currency = "USD",
                            total = FormatPrice(totalPrice)
                        },
                        item_list = new ItemList()
                        {


                            items = cart.ProductsInCart.Select(x => new Item()
                            {
                                name = x.Product.Name,
                                currency = "USD",
                                quantity = Convert.ToString(cart.
ProductsInCart.SingleOrDefault(i => i.Product.Id == x.Product.Id).Quantity
),
                                price = FormatPrice(x.Product.Price)
                            }).ToList()

                        }
                    }
                },
                redirect_urls = new RedirectUrls
                {
                    return_url = Url.Action("Return", "Payment", null, Request
.Url.Scheme),
                    cancel_url = Url.Action("Cancel", "Payment", null, Request
.Url.Scheme)
                }
            };

            // Send the payment to PayPal
            var createdPayment = payment.Create(apiContext);

            /* code for debugging paypal requests
            try
            {
                var createdPayment = payment.Create(apiContext);
```

```
71              }
72              catch (PayPal.PaymentsException ex)
73              {
74                  Response.Write(ex.Response);
75              }
76              */
77
78              // Save a reference to the paypal payment
79              orderInfo.PayPalReference = createdPayment.id;
80              _context.SaveChanges();
81
82              // Find the Approval URL to send our user to
83              var approvalUrl =
84                  createdPayment.links.FirstOrDefault(
85                      x => x.rel.Equals("approval_url", StringComparison.
    OrdinalIgnoreCase));
86
87              // Send the user to PayPal to approve the payment
88              return Redirect(approvalUrl.href);
89          }
90          return View("Index");
91      }
```

Kod 7: **PaymentAction**

PaymentAction najprije provjeri je li je model koji prima iz forme (vidi slika 8) valjan, to jest je li su zadovoljeni uvjeti validacije. Zatim kreira objekt **orderInfo** koji predstavlja instancu klase **OrderInfo** i u njega sprema podatke o narudžbi primljenje iz forme i nakon toga sprema tu narudžbu u bazu podataka. Zatim je potrebno kreirati **apiContext** objekt pomoću kojeg se obavlja komunikacija s PayPal-om. Za to će poslužiti pomoćna funkcija **GetApiContext()** (vidi kod 8) koja iz **web.config** dohvaća podatke za pristup PayPal aplikaciji (vidi kod 5) i pomoću njih dolazi do **AccesssToken-a** pomoću kojeg se kreira novi **ApiContext** objekt.

```
public static APIContext GetApiContext()
        {
            // Authenticate with PayPal
            var config = ConfigManager.Instance.GetProperties();
            var accessToken = new OAuthTokenCredential(config).GetAccessToken
    ();
            var apiContext = new APIContext(accessToken);
            return apiContext;
        }
```

Kod 8: **GetApiContext()** funkcija

Sada treba kreirati **payment** objekt koji služi za slanje informacija o kupnji PayPal-u. Objekt **payment** definira se koristeći PayPal Payments REST API. Payments *prostor imena* (eng. namespace) uključuje resurse za plaćanje, prodaju, povrat novca, autorizaciju i narudžbe. **Payment** objekt sadrži sljedeće atribute (vidi kod 7):

- **intent** – unutar ovog atributa definira se namjera plaćanja i obavezan je(eng. required). Može poprimiti tri vrijednosti:

  „**sale**“ – plaćanje se vrši trenutno,

  „**authorize**“ – plaćanje se autorizira odmah, ali se ne naplaćuje odmah nego u neko trenutku u budućnosti,

  „**order**“ - plaćanje se odmah autorizira, a naplata se vrši u roku 29 dana,

- **payer** – govori odakle dolaze sredstva za plaćanje. Najčešće se koristi vrijednost „paypal“ koja govori da se kupca šalje na PayPal i on tamo može koristiti bilo koji oblik plaćanja podržan na njegovom PayPal računu. Umjesto „paypal“ vrijednosti možemo koristi informacije o bankovnom računu,

- **transactions** – lista objekata **transaction**. Unutar objekta **transaction** definira se za koga je plaćanje i tko vrši plaćanje za svaku pojedinu transakciju. Jedna transakcija može sadržavati više proizvoda. Sadrži sljedeće atribute:

  - **amount** – obavezan (eng. required) atribut. Sadrži informacije o količini novca potrebnog za plaćanje. Ima dva obavezna atributa i jedan opcionalan atribut:

    * **currency** – valuta koja se koristi za plaćanje. Koristi se ISO-4217 zapis s tri znaka za prikaz valute,

        * **total** – služi za spremanje iznosa novca potrebnog za plaćanje transakcije. Iznos se sprema u obliku riječi (eng. string) maksimalne duljine od deset znakova na sljedeći način: sedam znakova ispred decimalne točke, decimalna točka i dva znaka nakon decimalne točke,
        * **details** – objekt u kojeg se spremaju dodatne informacije vezane za plaćanje poput cijene poštarine i poreza,
    - **reference_id** – opcionalan atributa koji trgovac može dati svakoj transakciji,
    - **payee** – sadrži informacije vezane za osobu koja prima novac i izvršava narudžbu,
    - **description** – opis transakcije,
    - **note_to_payee** – napomena za primatelja novca,
    - **custom** – polje u koje klijent može koristiti kako bi napisao svoje napomene,
    - **invoice_number** – broj dostavnice kako bi se lakše pratio proizvod,
    - **purchase_order** – broje narudžbe ,
    - **soft_descriptor** – kratki kod kojim je opisana kupovina do 22 znaka,
    - **payment_options** – opcije plaćanja vezane za ovu transakciju,
    - **notify_url** – URL na koji se šalje obavijest o plaćanju,
    - **order_url** – URL na e-trgovini vezan za ovo plaćanje, prikazuje informacije o narudžbi,
    - **related-resources** – lista transakcija vezanih za plaćanje. Transakcija definira za koga je plaćanje i tko vrši plaćanje. Ovaj lista je *samo za čitanje* (eng. read only),
    - **item_list** – sadrži listu items unutar koje se spremaju svih proizvodi (eng. item) koji su kupljeni unutar određene transakcije. Lista items sastoji se od objekata Item koji sadrže atribute poput name, quantity, price, currency koji služe za spremanje informacija vezanih za određeni proizvod (eng. item). **Item_list** još dodatno sadrži i atribute **shipping_address**, **shipping_method** i **shipping_phone_number** koji služe za spremanje informacija vezanih za dostavu prozivoda,

- **experience_profile_id** – ID generiran od strane PayPal-a, služi za dodavanje i uklanjanje određenih opcije plaćanja. Na primjer ako se prodaje+u samo digitalne stvari

nije potrebno imati opciju dosta,

- **note_to_payer** - polje unutar kojeg se spremaju napomene kupcu,
- **redirect_url** - objekt koji se sastoji od dva URL-a koji služe za preusmjeravanje kupca nakon obavljene ili neuspješne kupovine. Oba URL-a su obavezna kod plaćanja putem PayPal-a.
  - **return_url** - URL na koji je kupac preusmjeren nakon što odobri plaćanje
  - **cancel_url** - URL na koji je kupac preusmjeren nakon što otkaže plaćanje

Nakon kreiranja **payment** objekta on se šalje PayPal servisu kako bi on kreirao taj isti objekt. Odgovor od PayPal-a je isti taj objekt s dodanim atributima **id**, **create_time**, **update_time**, **state** i **links**. ID dobiven od strane PayPal-a sprema se u kreirani **orderInfo** objekt te se promjene na tom objektu spremaju u bazu podataka. Preostaje definirati URL na koji će se kupca preusmjeriti kako bi platio narudžbu putem PayPal-a. Bitno je primjetiti da postoje tri URL-a u PayPal-ovom odgovoru (vidi kod 9):

- **self URL** – koristi se kako bi se dobio sam **payment** objekt
- **execute URL** – koristi se kako bi se izvršilo plaćanje
- **approval_url** – koristi se za slanje kupca na stranicu PayPal-a

Self i **execute** URL-ove može ponovno kreirati sam trgovac koristeći ID **payment**-a.

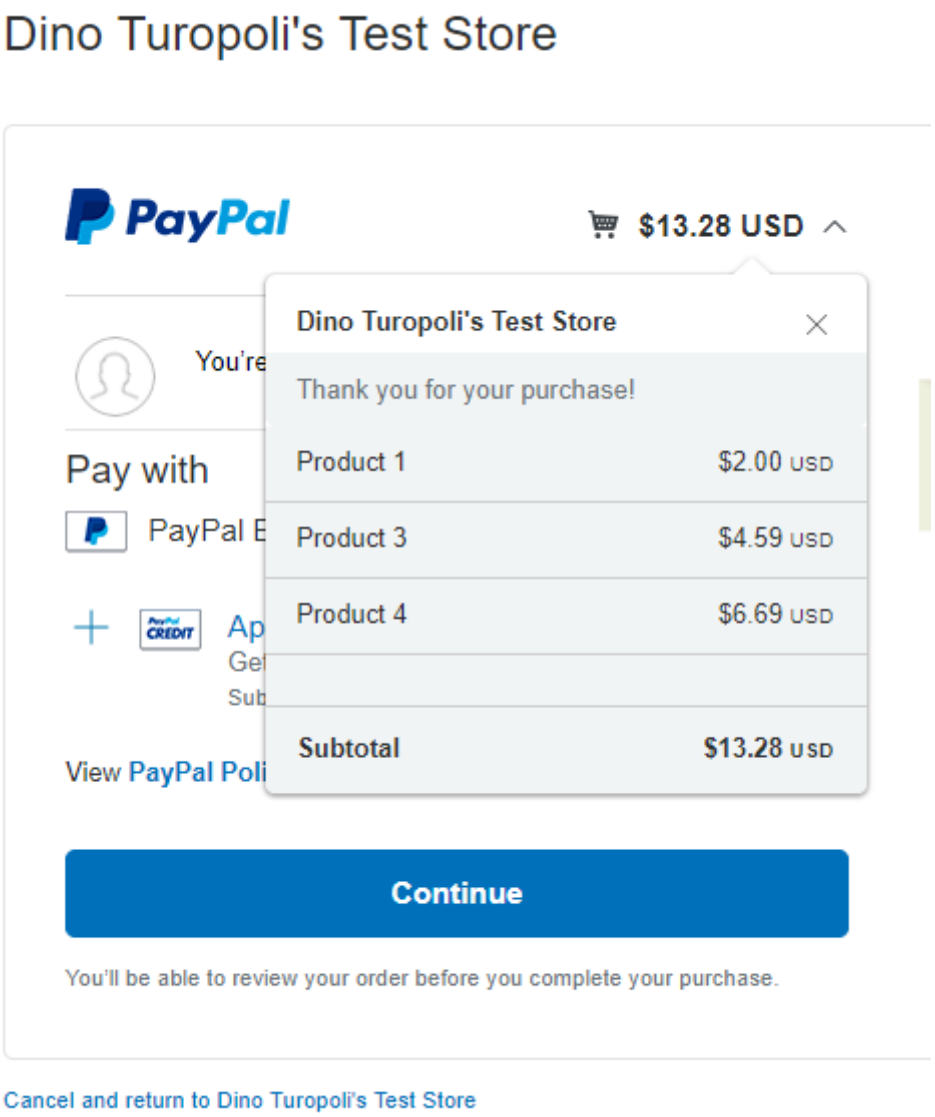
```
{
  "id": "PAY-1B56960729604235TKQQIYVY",
  "create_time": "2017-09-22T20:53:43Z",
  "update_time": "2017-09-22T20:53:44Z",
  "state": "created",
  "intent": "sale",
  "payer": {
    "payment_method": "paypal"
  },
  "transactions": [
    ...
  ],
```

```
  "links": [
    {
      "href": "https://api.sandbox.paypal.com/v1/payments/payment/PAY-1B56960729604235TKQQIYVY",
      "rel": "self",
      "method": "GET"
    },
    {
      "href": "https://www.sandbox.paypal.com/cgi-bin/webscr?cmd=_express-checkout&token=EC-60385559L1062554J",
      "rel": "approval_url",
      "method": "REDIRECT"
    },
    {
      "href": "https://api.sandbox.paypal.com/v1/payments/payment/PAY-1B56960729604235TKQQIYVY/execute",
      "rel": "execute",
      "method": "POST"
    }
  ]
}
```

Kod 9: Primjer PayPal-ovog odogovora nakon kreiranja **payment** objekta. Prikazani su samo atributi koje PayPal dodaje

Nakon što kupac popuni formu i klikne na gumb za potvrdu plaćanja (vidi slika 8). Kreira se **payment** objekt i preusmjeren je na stranicu PayPal-a kako bi potvrdio plaćanje. Prilikom potvrde plaćanja na PayPal servisu kupac još jednom može pogledati informacije vezane za njegovu narudžbu. Ako nije sve u redu i kupac želi odustati od narudžbe pri dnu ima gumb kojim odustaje od narudžbe (vidi slika 9). Klikom na odustajanje aktivira se **cancel_url** koji poziva **Cancel** akciju u kontroleru i preusmjerava kupca na pogled koji ga upozorava da je otkazao narudžbu. Ako kupac želi potvrditi narudžbu onda klikom na nastavak obrade plaćanja (vidi slika 9) aktivira **return_url** koji poziva **Return** akciju unutar kontrolera. Return akcija prima dva argumenta **payerId** i **paymentId** pomoću kojih će se izvršiti plaćanje. Najprije se dohvaća narudžba iz baze podataka na osnovu **paymentId**-a te se definira **apiContext**. Sada se definiraju dva objekta, **paymentExecution** objekt unutar kojeg se definira platitelj pomoću **payerId**-a i **payment** objekt gdje se pomoću **paymentId**-

a definira za koju narudžbu se želi izvršiti plaćanje. Pomoću **Execute()** funkcije završava se plaćanje i kupac se preusmjerava na **ThankYou** akciju koja poziva pogled koji zahvaljuje kupcu na njegovoj kupovini. (vidi kod 10)



Slika 9: Primjer završavanja kupovine na PayPal servisu

```
1 public ActionResult Return(string payerId, string paymentId)
2         {
3             var order = $\_$context.ProductOrderInfos.FirstOrDefault(x => x.
    PayPalReference == paymentId);
4 
5             var apiContext = GetApiContext();
6 
```

```
7             // Set the payer for the payment
8             var paymentExecution = new PaymentExecution()
9             {
10                payer_id = payerId
11            };
12
13            // Identify the payment to execute
14            var payment = new Payment()
15            {
16                id = paymentId
17            };
18
19            // Execute the Payment
20            var executedPayment = payment.Execute(apiContext, paymentExecution
    );
21
22            return RedirectToAction("ThankYou");
23        }
```

Kod 10: **Return** akcija

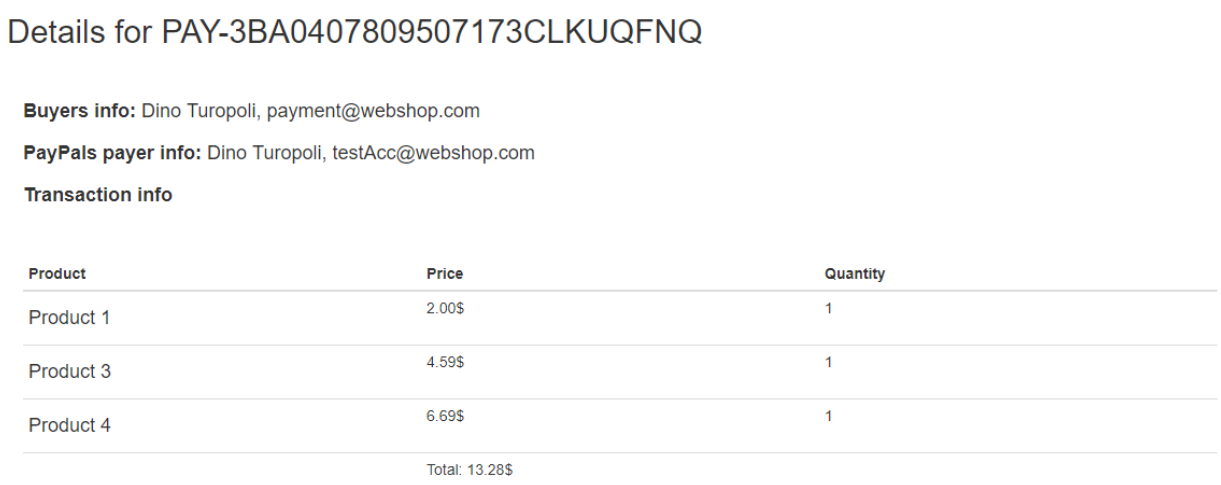
Nakon završetka transackije račun za svaku pojedinu transakciju sprema se na PayPal-ovom servisu. Također administrator e-trgovine u svakom trenutku može pristupiti podacima vezanima za svaku transakciju na stranici same e-trgovine. Na navigacijskoj traci nalazi se poveznica **Sales** koja vodi administratora na pogled vezan za obavljene transakcije. Tamo admin može vidjeti ID transakcije, vrijeme kreiranja, iznos novca i status transakcije. Također administrator ima i opciju povrata (eng. refund) transakcije. (vidi slika 10)

List of made sales

Payment ID	Created Date	Payment Status	Total	Sale Status	
PAY-0UN28945W6258884ALKRRDNA	2018-03-09T22:59:00Z	approved	12.00$	completed	Refund
PAY-3BA0407809507173CLKUQFNQ	2018-03-14T11:08:38Z	approved	13.28$	completed	Refund
PAY-4TU44435A5882044HLKUQSLY	2018-03-14T11:36:15Z	created	58.46$	Canceled!	
PAY-4JB22414RH4648350LKURAIY	2018-03-14T12:05:55Z	approved	2.00$	completed	Refund
PAY-9NG073193A729594GLKURBBY	2018-03-14T12:07:35Z	approved	17.28$	completed	Refund

Slika 10: Lista obavljenih transakcija

Klikom na ID transakcije mogu se dobiti detaljnije informacije vezane za transakciju. Administrator može vidjeti podatke koje kupac upisuje na e-trgovini prilikom ispunjavanja forme za kupovinu, informacije vezane za PayPal račun putem kojeg je obavljena transakcija te listu proizvoda kupljenih u transakciji. (vidi slika 11)



Slika 11: Informacije vezane za transakciju

5 Zaključak

Pojavom e-trgovine znatno se ubrzava i olakšava proces kupovine proizvoda i omogućuje trgovcima širenje njihovih tražišta na globalnoj razini. Koriste bezgotovinski način plaćanja i usmjerivač plaćanja te zbog toga kupac i trgovac ne moraju ostvariti fizički kontakt niti u jednom trenutku procesa kupovine. Najraširenij model elektroničkog plaćanja je plaćanje platnim karticama. Usmjerivač plaćanja brine se za sigurnost prijenosa osjetljivih podataka između banke zamprimatelja i banke izdavatelja kartice. Sigurnost je osigurana PCI DSS standardom koji traži od trgovca da se pridžava određnih sigurnosnih zahtjeva. Zbog svega navedenog e-trgovine predstavljaju sadašnjost i budućnost kupovine i razmjene proizvoda.

Koristeći razvojni alata Visual Studio i ASP.NET MVC tehnologiju razvio sam prototip e-trgovine s implementiranim usmjerivačem plaćanja za PayPal Express Checkout servis.

Literatura

[1] V. Prakash Gulati, S. Srivastava , *The Empowered Internet Payment Gateway*, 2015

[2] E-commerce, https://www.2checkout.com/ecommerce-glossary

[3] E-commerce tutorial, https://www.tutorialspoint.com/e_commerce/index.htm

[4] M. Hamendi, *The Complete ASP.NET MVC 5 Course*, https://www.udemy.com/the-complete-aspnet-mvc-5-course/

[5] PayPal documentation, https://developer.paypal.com/docs/api/payments/

[6] PCI Security Standards Council, *Requirments and Security Assessment Procedures*, version 3.2, 2016, https://www.pcisecuritystandards.org/documents/PCI$_$DSS$_$v3-2.pdf?agreement=true&time=1519483306922

Sažetak

Ovaj diplomski rad opisuje teoriju vezanu za e-trgovinu, model plaćanja platnim karticama i problematiku usmjerivača plaćanja. Pojavom interneta počinje razvoj e-trgovina putem kojih se ubrzava i olakšava proces kupovine. Svaka e-trgovina ima implementira neki oblik usmjerivača plaćanja i koristi određeni model elektroničkog plaćanja. Najpopularniji model elektroničkog plaćanja je plaćanje platnim karticama. Bitnu ulogu u plaćanju platnim karticama igra usmjerivač plaćanja koji služi kao sigurni most za komunikaciju između kupca i trgovca preko njihovih banaka. Usmjerivač plaćanja mora se pridržavati PCI DSS sigurnosnog standarda. U praktičnom dijelu rada izraditi ćemo prototip e-trgovine sa ugrađenim usmjerivačem plaćanja za PayPal Express Checkout servis. Aplikacija je razvijena uz pomoć Adacta-e.

Ključne riječi: e-trgovina, platna kartica, usmjerivač plaćanja, PayPal, ASP.NET MVC

Summary

This master's thesis describes theory of e-commerce, payment card payment model and payment gateway problematic. With the appearance of the internet begins the development of e-commerce which facilitates payment process. Every e-commerce application has implemented some form of payment gateway and uses certain form of electronic payment. The most popular electronic payment model is payment card model. Payment gateway plays important role in a payment card payment model. It acts like a secured bridge for communication between issuing and acquiring banks. Payment gateway must comply with the PCI DSS security standard. In the end of this master's thesis we will develop a prototype of e-commerce with a built-in payment gateway for PayPal Express Checkout service. Application is developed in cooperation with Adacta.

Key words: e-commerce, payment card, payment gateway, PayPal, ASP.NET MVC

Životopis

Dino Turopoli rođen je 19. ožujka 1994. u Virovitici. Školovanje započinje u Osnovnoj školi Josipa Kozarca u Slatini, a nakon završene osnovne škole upisuje opću gimnaziju Srednje škole Marka Marulića u Slatini. 2012. godine upisuje prediplomski studij matemtaike na Odjelu za matematiku Sveučilišta Josipa Jurja Strossmayera u Osijeku. Završava predimplomski studij 2015. godine s završnim radom na temu „SSH protokol“ pod vodstvom mentora izv. prof. dr. sc. Domagoja Matijevića te iste godine uspije diplomski studij matematike, smjer Matematika i računarstvo. Tijekom studiranja sudjeluje na programerskom natjecanju IEEEXtreme.