Vježba 9 - MySQL i Node

U ranijim vježbama smo mogli vidjeti kako je rad sa datotekama kao jednim vidom perzistencije podatka prilično komplikovan. Iako datoteke mogu biti u poznatom i standardnom formatu i podaci u njima mogu biti pravilno struktuisani još uvjek postoje stvari koje nisu rješene poput: konkurentnog upisivanja u datoteke, relacije između podataka, ažuriranje podataka koji su zavisni jedni od drugih i sl. Kako bi riješili navedene probleme koristit ćemo MySQL bazu podataka.

Povezivanje MySQL baze podataka sa Node aplikacijom se ostvaruje korištenjem **mysql** paketa. Konekciju na bazu kreiramo sa **mysql.createConnection** i kao rezultat dobijamo connection objekat. Nad ovim objektom je moguće izvršavati upite kao da direktno radimo sa bazom podataka. Pri kreiranju konekcije navode se parametri: host, port, user, password, database, charset, timezone i sl.
Prije prvog upita potrebno je pozvati metodu **connection.connect()**, a nakon posljednjeg upita potrebno je zatvoriti konekciju sa **connection.end()**. Obje metode primaju kao parametar callback funkciju koja sadrži podatke o greški u svom parametru.
Ovaj paket ima i podršku za connection pooling. Connection pooling je skup konekcija koje su otvorene prema bazi i koje se mogu ponovo iskoristiti pri različitim upitima. Koristi se kako se ne bi morale otvarati/zatvarati konekcije za svakog korisnika jer taj proces usporava rad aplikacije. Pool se kreira pozivanjem **mysql.createPool()** gdje se kao parametar proslijedi objekat sa connection string parametrima (host, port, username, …). Iz kreiranog poola konekciju dobijamo sa **pool.getConnection()**, ova funkcija kao parametar prima callback koji sadrži dobivenu konekciju ili ako se desila greška informacije o istoj.
Upite nad konekcijom pozivamo sa **connection.query**, gdje kao prvi parametar prosljeđujemo string upita, a drugi je callback. Callback sadrži podatke o grešci, rezultatu upita i treći parametar daje dodatne informacije o kolonama iz upita.

Za više detalja pogledajte dokumentaciju.

Zadatak 1. (obavezan)
Kreirajte bazu podataka koristeći phpmyadmin koji je instaliran na računarima (phpmyadmin se nalazi na http://localhost/phpmyadmin/index.php username:root, password:password). Baza treba da ima jednu tabelu imenik koja ima sljedeće kolone: ime i prezime varchar(30), adresa varchar(30) i broj telefona varchar(20).

Zadatak 2. (obavezan)
Kreirajte Node aplikaciju koja će kada se uputi GET zahtjev na **localhost:3000/imenik** vratiti html tabelu sa svim zapisima iz imenika.

Zadatak 3. (obavezan)
Kreirajte formu kojom se mogu unositi podaci u tabelu imenik. (Možete realizovati putem Ajax-a ili korištenjem html formi)

Zadatak 4. (obavezan)
Kreirajte novu tabelu adresar koja će sadržavati idKontakta - veza na red iz tabele imenik i idPoznanik - veza na red iz tabele imenik.

Zadatak 5. (obavezan)
Dodajte novu rutu u aplikaciju koja će odgovarati na GET zahtjev **localhost:3000/poznanik/:kontakt.** Odgovor na ovaj zahtjev treba biti html tabela sa svim kontaktima (ime, adresa i broj telefona) iz adresara kojima je idKontakta isti kao **:kontakt** iz url-a (poznanici kontakta sa id-em kontakt).

Kada završite zadatke provjerite:

- Da li je vaša aplikacija podložna cross-site scripting (XSS) napadima? Može li se unijeti vrijednost koja sadrži HTML ili JavaScript kod? Šta ćete učiniti da to spriječite?
- Perzistentni XSS je posebno opasan oblik XSS napada. Do njega dolazi kada je napadač u stanju da ubaci maliciozan HTML ili JavaScript kod u bazu, tako da svaki sljedeći posjetilac vidi ovaj kod. Na taj način napadač se može baviti i stvarima kao što je prikupljanje šifara (koristeći JavaScript može se dodati event listener na polje forme i zatim poslati na udaljeni web servis).
- SQL injection je napad koji je po tipu sličan XSS. U SQL injection propustima postoje upiti koji kao parametar uzimaju vrijednost koju je korisnik unio (pretraga redova po nekoj vrijednosti koju je korisnik unio i sl.). Zadatak 5 ima potencijalno mjesto gdje se ovaj propust može pojaviti. Identifikujte problem i probajte iskoristiti navedeni propust. Kod napada koji koristi SQL injection propust zlonamjerni korisnik u korisnički unos upisuje SQL kod koji može nanijeti štetu aplikaciji (obrisati redove, izmijeniti neke vrijednosti i sl.). Više o ovom propustu na OWASP.
- Na koji način je moguće spriječiti SQL injection? (Pročitajte u dokumentaciji mysql modula! Šta je preporuka autora ovog modula?)