ELABORAT ZAVRŠNOG RADA

SUSTAV ZA VOĐENJE ŠKOLSKE INVENTURE

Mentor: Mirko Jambrošić

Učenik: Vito List 4RT

1. UVOD

1.1. OPIS ZADATKA

Inventura je proces evidentiranja, praćenja i dokumentiranja ukupne imovine ili zalihe koje posjeduje neka ustanova, u ovom slučaju škola. Cilj ovog postupka je točno utvrđivanje količine, vrijednosti i stanja inventara u nekom trenutku. Obično se provodi periodično, npr. jednom na godinu kako mi se osiguralo da evidencija inventara odgovara stvarnom stanju. No kod klasičnog provođenja inventure moguće su ljudske pogreške tijekom izrade izvješća te je korištenje sistema koji vodi brigu o tome puno efikasnije.

Zadatak je bio napraviti sustav za vođenje školske inventure. Sustav omogućuje kreiranje korisnika od kojih je jedan administrator te ima dodatna ovlaštenja. Rješenje omogućuje dodavanje proizvoda i učionica te ostale parametre potrebne za knjigovodstvo. Korisnik može koristiti ugrađen skener za barkod kako bi u određenoj učionici skenirao proizvode te ih tako evidentirao u trenutnu inventuru, najbolje koristiti na mobilnom uređaju. Prije početka skeniranja proizvoda važno je skenirati barkod učionice kako bi se naznačilo kojoj prostoriji pripada proizvod. U slučaju neispravnog barkoda korisnik je obaviješten porukom te mu se nude drugi načini evidentiranja. Administrator može pratiti aktivnost pojedinog korisnika te odobriti promjene dođe li do njih, ali tek nakon što inventura završi (npr. promjena učionice proizvoda). Na kraju inventure dobiva se jasno izvješće koje prikazuje proizvode te njihove količine po učionicama te je li došlo do promjena.

1. TEORIJSKI DIO

2.1. KORIŠTENI ALATI

Za izradu rješenja korišten je isključivo softver. Od programskih jezika odabrao sam PHP, HTML, CSS, JavaScript, Python, MySQL te Apache Server za *hosting* aplikacije.

2.2. ULOGA POJEDINOG ALATA

PHP: Koristio sam PHP za dinamičko generiranje web stranice te komunikaciju sa bazom. Najbolji primjer bio bi vađenje podataka o proizvodima i generiranje tablice. U PHP-u se također odvija *routing* koji nam omogućuje *user-friendly* linkove (npr. inventura/admin umjesto inventura/views/admin.php). Također ga koristim za generiranje barkodova na strani servera za nove proizvode.

HTML: Koristi se unutar PHP datoteke. Definira sadržaj koji korisnik vidi.

CSS: Najveći dio CSS-a odrađen je korištenjem *Bootstrap* klasa. *Bootstrap* je biblioteka otvorenog koda koja nam pruža već definirane stilove za klase. Tako naš UI izgleda dosljedno na svakoj stranici te je minimalistički. Također sam koristio i svoje stilove za stvari za koje nam *Bootstrap* ne pruža.

JavaScript: Uglavnom za dinamičnost web stranice sa korisnikove strane i da slanje POST zahtjeva prema serveru da ažuriranje stanje u bazi. Koristio sam *JQuery*, biblioteka, koja nam olakšava rad sa HTML elementima u JavaScriptu.

Python: Njegova jedina uloga u ovom projektu bila je popunjavanje baze podataka sa probnim podacima kako bi lakše testirali aplikaciju na većem broju zapisa.

MySQL: Kao bazu podataka odabrao sam MySQL. Jednostavan je za korištenje te se lako integrira sa PHP-om i Python-om. On je tip jezika za izradu relacijskih baza podataka.

Apache Server: koristim da za *routing* tj. *user-friendly* linkove

3. IZRADA RJEŠENJA

Prije same izrade ovog rješenja potrebno je dobro definirati bazu podataka jer nam ona određuje velik dio strukture programa, te po potrebi konfigurirati web server.

3.1. KONFIGURACIJA APACHE WEB SERVERA

Kako bi radio prije spomenuti *routing* ili preusmjeravanje morao sam kreirati *.htaccess* datoteku u mapi projekta kako bi konfigurirao Apache Server. Ondje sam naveo da sve zahtjeve preusmjeri na index.php datoteku osim ako nije navedena određena datoteka, na primjer slike.

|  |
| --- |
| <IfModule mod\_rewrite.c>  RewriteEngine On  RewriteCond %{REQUEST\_FILENAME} !-f  RewriteRule ^ index.php [L]  </IfModule> |

Konfiguracijski kod u .htaccess datoteci.

3.2. ARHITEKTURA BAZE PODATAKA

Baza za ovu aplikaciju sastoji se od više tablica. Dobro je razdvojiti pojedini segment ili značajku aplikacije u svoju tablicu radi preglednosti i lakšeg upravljanja.

TU IDE SLIKA RELACIJI BAZE

3.2.1. OPIS TABLICA

S obzirom da aplikacija omogućuje registraciju i prijavu korisnika za to je predviđena tablica **users**, ondje se spremaju osobni podaci za svakog korisnika te najvažnija stvar – njegova uloga, admin ili običan korisnik.

Nakon svake prijave u bazu se sprema *session cookie* od korisnika koju mu omogućuje prijavu na dulje vrijeme čak i kad se zatvori preglednik, za to se koristi tablica **session\_cookies**.

Radi boljeg definiranja i kasnije sortiranja proizvoda važno je bilo napraviti tablicu **tipovi\_proizvoda**, ovu tablicu preko aplikacije popunjava admin.

Jedna od važnijih tablica jest **proizvodi**, ona sadrži tip određenog proizvoda, njegovo ime te opis, ključne knjigovodstvene parametre te poveznicu na sliku, sliku je dobro koristiti kod proizvoda koje je teže identificirati preko samog imena.

Usko vezana tablica u proizvode jest **barkodovi**. Ondje svaki proizvod povezujemo sa njegovom vrijednosti barkoda. Za razlikovanje proizvoda od učionice, vrijednost proizvoda započinje znamenkom 1.

Tablica **ucionice** sadrži oznaku te barkod svake učionice koje admin doda, barkod učionice dogovoreno uvijek počinje znamenkom 2 kako ne bi došlo do kolizije sa već postojećim barkodom proizvoda te radi lakšeg prepoznavanja.

Za vrijeme trajanja inventure svaki skenirani proizvod sa odgovarajućom učionicom i korisnikom koji je obavio evidenciju zapisuje se u tablicu **evidencija**, u slučaju krivog skeniranja admin može poništiti evidenciju te se ona kasnije neće uzimati u obzir kod ažuriranja stanja.

Kako je moguće pokrenuti više inventura, za to je predviđena tablica **inventura**, ona samo sadrži ID te početak i kraj svake inventure, datum završetka koristi se kod prikaza stanja kako bi se korisniku dao do znanja zadnji datum ažuriranja.

U tablici **stanje** povezani su proizvodi sa odgovarajućim učionicama te je navedena količina. Ona se ažurira nakon završetka inventure.

Neobavezna tablica koja se nalazi u bazi je **mysql\_errors**, ona je korištena isključivo u svrhe *debugiranja* te nikako ne utječe na rad ili korisničko sučelje aplikacije. Njena jedina korist je zapisivanje grešaka koje su nastale u ključnim dijelovima aplikacije sa odgovarajućim datumom koja kasnije doprinosi lakšem održavanju.

3.3. ZNAČAJKE

S obzirom da inventura sama po sebi zahtjeva dosta posla i organizacije, ovo rješenje nudi mnogo značajki koje bi što realističnije rekreirale to u digitalnom obliku.

Prije bilo kakvih aktivnosti korisnik se mora registrirati. Ondje navodi svoje ime, prezime, e-mail adresu čija je upotreba predviđena za buduće širenje aplikacije, korisničko te lozinku. Nakon uspješno izrađenog profila, e-mail adresa mora zadovoljavati uvjete i korisničko ime je dostupno, korisnik odlazi na stranicu prijava te koristi podatke za prijavu. Sve se lozinke *hashiraju* tijekom upisa u bazu podataka kako bi se osigurala sigurnost podataka i spriječile neželjene prijave.

Najbitnija značajke ove aplikacije jest barkod skener, on je dostupan svakom korisniku na početnoj stranici koja se otvara nakon prijave. Za izradu skenera korištene su funkcije iz javne JavaScript biblioteke pod nazivom ZXing. Kada korisnik ode na stranicu za skeniranje nudi mu se odabir prednje ili stražnje kamere. Kad je kamera odabrana i upaljena pokreće se kod koji konstantno analizira ulaz iz kamere kako bi prepoznao barkod. Kada se prepozna barkod njegova se vrijednost šalje na server te se ondje obrađuje. Ukoliko se radi o barkodu učionice vrijednost se vraća korisniku te se kasnije koristi ako se skenira proizvod. Ako korisnik skenira proizvod, a da nije prethodno skenirao učionicu o tome je obaviješten. Ako je skeniranje uspješno ono se sa trenutnim vremenom i korisnikom zapisuje u odgovarajuću tablicu.

Da bi mogli išta skenirati postoji opcija dodavanja proizvoda koji ima samo administrator sustava. Kada dodaje proizvod njemu dodjeljuje naziv te može dodati i opis uz njega. Također se uz to zahtjeva da odabere kojoj učionici pripada te kojeg je tipa proizvod. Dodavanje i uređivanje učionica i tipova također je dostupno samo administratoru. Uz to mora se odabrati količina top proizvoda za odabranu učionicu i knjigovodstvene vrijednosti. To je početna vrijednost te koeficijent tj. postotak koji se tijekom obračuna oduzima od trenutne vrijednosti. Radi lakšeg prepoznavanja proizvoda tu je i opcija dodavanje slike uz svaki proizvod što je velika prednost naspram ručnog vođenja inventure. Na istoj se stranici pojavljuje tablica sa svim proizvodima te njihovim podacima.

Kad se doda proizvod u bazu uz njega se generira i barkod te slika barkoda u svg formatu. Za to se koristi također biblioteka ZXing ali prilagođena za php jezik kako bi je pokretali na strani poslužitelja. Generirani se barkodovi spremaju u svoju mapu te se prikazuju u aplikaciji. Kako postoji više formata barkodova za ovo je rješenje korišten EAN13 format. On se sastoji od 13 znamenki, 12 znamenki određujemo mi, a zadnja je kontrolna kako bi se povećala sigurnost i preciznost tijekom kodiranja. Isti se proces izvodi i kod dodavanja učionice.

Kad korisnik pregledava evidenciju nude mu se različiti filteri. Može birati učionicu, korisnika koji je evidentirao, tip proizvoda, identifikacijski broj inventure te pretraživati prema nazivu proizvoda.

Na stranici *stanje* vidi se trenutno stanje svakog proizvoda po količini u pripadajućoj učionici. Ondje se ništa ne može mijenjati, jedina je opcija izrada PDF dokumenta koja ispisuje tablicu sa stanjem u datoteku u A4 formatu. Kod odabira ispisa u PDF može odabrati samo opcija ispisa barkodova koji se mogu koristiti za obilježavanje proizvoda.

Svaku inventuru pokreće administrator. Na stranici *usporedba* može vidjeti razlike između stanja proizvoda i trenutne evidencije te prema potrebi otpisati pojedini proizvod iz učionice ili potvrditi promjenu učionice ako je došlo do nje.

Svaki korisnik, običan ili administrator, ima pristup stranici svog profila. Ondje su navedeni njegovi podaci koje unosi tijekom registracije uz što je još dodano polje gdje može vidjeti svoju ulogu te datum izrade profila.

4. ARHITEKTURA APLIKACIJE

Radi lakšeg snalaženja i organizacije datoteka, kod je pisan po uzoru na MVC (engl. *model-view-controller)* arhitekturu što nam omogućuje da razdvojimo logiku za svaku stranicu od sučelja koje korisnik vidi. Za svaku stranicu postoji *controller* PHP datoteka u kojoj su definirane funkcije za dobivanje podataka iz baze podataka te obrada GET i POST zahtjeva. Uz controller postoji i *view* datoteka u koji je definirano sučelje za korisnika te je potpuno odvojeno od logike, sve potrebne varijable inicijalizirane su i controller datoteci te ih samo prikazujemo u view-u. Da bi bila prava MVC arhitektura trebali bi i uključiti *model* datoteke ali radi manje kompleksnosti projekta to je ukomponirano u controller datoteke. Sve controller i view datoteke nalaze su odgovarajućim mapama radi organiziranosti.

4.1. POVEZIVANJE NA BAZU

Kao baza podataka koristi se MySQL. To je sistem da održavanje i izradu relacijskih baza podataka. Na bazu se spajamo na strani poslužitelja kako bi pridobili potrebne podatke. Poslužiteljski jezik PHP nam nudi gotove funkcije za spajanje i izvršavanje SQL upita. Za aplikaciju je izrađena klasa MySQLDB. Ondje su definirane metode *select*, *delete*, *insert* i *update* kako bi se smanjila količina koda pri korištenju baze u ostalim datotekama. Na samom početku programa u index.php datoteci inicira se objekt tipa novoizrađene klase. Nije korišten *singleton* jer sve ostale controllere uključujemo (engl. include) na kraju index.php datoteke te je možemo dalje koristiti bez potrebe za ponovnom inicijalizacijom objekta.