1. **ŽILINSKÁ UNIVERZITA V ŽILINE**
2. FAKULTA ELEKTROTECHNIKY A INFORMAČNÝCH TECHNOLÓGIÍ
3. Katedra riadiacich a informačných systémov
4. **Semestrálna práca: Fakturačný systém**
5. **Meno a priezvisko: Peter Leško**
6. **Študijná skupina:** **3ZKA1A**  
   **Dátum: 23.4.2025**

Obsah

[Úvod a zadanie 1](#_Toc196572273)

[1 Teoretický úvod 2](#_Toc196572274)

[1.1 Databáza MySQL 3](#_Toc196572275)

[1.2 PHP 3](#_Toc196572276)

[2 Praktická realizácia 3](#_Toc196572277)

[2.1 Návrh databázy 3](#_Toc196572278)

[2.1.1 Entitno-relačný diagram 4](#_Toc196572279)

[2.1.2 Relačný dátový model 7](#_Toc196572280)

[2.2 Návrh systému 8](#_Toc196572281)

[2.2.1 Časť spolupracujúca s databázou – backend 9](#_Toc196572282)

[2.2.2 Časť interagujúca s užívateľom – frontend 12](#_Toc196572283)

[Záver 15](#_Toc196572284)

Úvod a zadanie

V dnešnej dobe je fakturačný softvér jedným zo základných blokov systémov používaných vo firmách na uľahčenie administratívnej práce. Je vytvorených niekoľko nástrojov pre vytváranie a správu úradných dokumentov, tým pádom aj faktúr. No väčšina z nich, ak nie všetky, má svoje používanie spoplatnené mesačným alebo ročným predplatným. Pričom funkcie týchto systémov siahajú aj ďalej než len na vytváranie a správu faktúr. Častokrát ponúkajú služby spojené aj s inými administratívnymi dokumentmi, ktoré veľké množstvo malých podnikateľov a živnostníkov väčšinou nevyužíva resp. ich vôbec nepotrebuje.

Preto úlohou tejto práce je navrhnúť jednoduchý softvér pre tvorbu a správu faktúr pre malých podnikateľov a živnostníkov. Ide o systém, kde nezaregistrovaný užívateľ si môže po zadaní všetkých potrebných údajov vytvoriť jednoduchú faktúru, bez nutnosti registrácie a následne vyexportovať do PDF. Tú si on uloží na vlastnom počítači, kde si vedie vlastnú inventúru. V prípade, že sa používateľ rozhodne zaregistrovať, jeho vytvorené faktúry spolu s jeho údajmi a zákazníkmi sa uložia do databázy a kedykoľvek k týmto údajom môže pristupovať. Takisto si bude môcť v prípade straty vytlačených faktúr ich opäť zo softvéru stiahnuť.

Požiadavky pre systém sú teda nasledovné:

* Registrácia a prihlásenie užívateľov
* Uloženie údajov o firme užívateľa
* Uloženie údajov o firme zákazníka užívateľa
* Vytvorenie, uloženie a správa faktúr
* Prehľad o vytvorených faktúrach a zákazníkoch

V nasledujúcich kapitolách si podrobnejšie uvedieme technológie, na ktorých postavíme tento systém a praktickú realizáciu daného softvéru.

# Teoretický úvod

Väčšina systémov pozostáva z troch prvkov:

* Databázy,
* Backendu, ako časti, ktorá spolupracuje s databázou a
* Frontendu, ako časti, ktorá interaguje s používateľom.

Databázu systém potrebuje, kvôli uloženiu systémových a používateľských údajov. V praxi sa využívajú rôzne databázové systémy, (napr. MongoDB (NoSQL), PostgreSQL, MySQL, ...). V našom prípade, keďže faktúra v zmysle dát by sa dala považovať aj ako slabo štruktúrovaná, by sme mohli použiť aj nejakú NoSQL databázu ako je MongoDB. No v našom prípade sme zvolili najčastejšie používanú relačnú databázu MySQL.

Backend je časť systému, ktorá prijíma požiadavky z frontendu, spracuje ich, odošle požiadovku vo forme SQL dopytu na databázu a naspäť odošle frontendu odpoveď na jeho požiadavku.

Frontend je časť systému, ktorá má za úlohu interagovať s používateľom, prijímať od neho vstupy, tie zvalidovať a odkomunikovať ich s backendom. Následne vypísať na obrazovku hlášky alebo vykonať požadovanú akciu. Táto časť kódu je priamo naviazaná s tým, čo sa nazýva „User experience“, čo definuje ako je užívateľ spokojný s používaním aplikácie alebo systému.

Programovací jazyk PHP je používaný aj pre backend aj pre frontend, pretože ako dokáže vytvárať dopyty a komunikovať s datábázou, tak dokáže aj dané výsledky prezentovať na webovej stránke. No jeho hlavné zameranie je spolupracovať s databázou (teda backend). Ako frontendový programovací jazyk nie je tak dobre vybavený ako napr. programovací jazyk JavaScript alebo jeho frameworky.

Rozdelenie systému na backend a frontend je relevantné v tom prípade, že používame rozdielne technológie pre ich vývoj a sú logicky oddelené, resp. vieme ukázať, kde sú hranice jedného a kde druhého. Takisto by sme mohli povedať, že jeden bez druhého vedia fungovať samostatne, resp. jeden aj druhý vieme nahradiť iným softvérom a celkovo systémy budú fungovať. Keďže v našom prípade na vývoj jedného aj druhého používame jeden programovací jazyk PHP, je uvažovanie o takomto rozdelení kúsok nepraktické. No v prípade, že v budúcnosti by sme frontend, naprogramovaný v PHP, nahradili kódom naprogramovaným napr. v jazyku JavaScript, toto rozdelenie by bolo plne relevantné. No to nám nebráni už v tomto bode rozmýšľať nad týmto logický oddelením systémov, kvôli prehľadnosti kódu a vzájomnej nezávislosti týchto častí na sebe.

## Databáza MySQL

MySQL je najčastejšie používaná relačná databáza. Dáta sú ukladané vo forme tabuliek, ktoré predstavujú jednotlivé entity. Jeden riadok tabuľky predstavuje jeden záznam danej entity a stĺpce tabuľky sú atribúty entity. Vzťahy medzi entitami sú definované cudzími kľúčmi v tabuľkách alebo samotnými prídavnými tabuľkami, ktoré spájajú dve alebo viacero tabuliek.

## PHP

PHP ako programovací jazyk je najčastejšou voľbou pri výbere jazyka, ktorý má spolupracovať s databázou. Ako programovací jazyk beží na servery, tj. keď príde dopyt od webového prehliadača na danú stránku, na strane servera zbehne celý program a výsledok od PHP sa pošle ako odpoveď webovému prehliadaču na jeho dopyt. Preto PHP nemá priamy prístup k interakcii s užívateľom pokiaľ nie je vytvorený dopyt od webového prehliadača na daný server.

# Praktická realizácia

Praktická realizácia pozostáva z návrhu databázy, naprogramovania časti systému, ktorá bude pristupovať komunikovať s databázou a časti systému, ktorá bude interagovať s používateľom. Následne sa môže databáza naplniť testovacími údajmi a otestovať funkčnosť celého systému.

## Návrh databázy

Správny návrh databázy spočíva vo veľkej miere na definovaní požiadaviek systému a ich správnej interpretácií. Požiadavky nášho systému spojené s databázou sú čiastočne spísané v úvode tejto práce takto:

* Možnosť registrácie a prihlásenia užívateľov
* Uloženie údajov o firme užívateľa
* Uloženie údajov o firme zákazníka užívateľa
* Vytvorenie, uloženie a správa faktúr
* Prehľad o vytvorených faktúrach a zákazníkoch

Len z prvého pohľadu na naše podmienky môžeme vidieť, že určite budeme potrebovať tabuľky (entity): *User* (Používateľ), *Company* (Firma) a *Invoice* (Faktúra). Pričom z dôvodu že nepoznáme presný počet faktúrovaných položiek, budeme potrebovať ďalšiu tabuľku *Ivoice\_item*, ktorá bude pomocou cudzieho kľúča pripojená k tabuľke *Invoice*. Takisto vieme oddeliť od tabuľky *Company* štruktúru, ktorá obsahuje údaje o adrese danej spoločnosti a tým dosiahneme väčšiu škálovateľnosť našej databázy, kedy je možné pridaním cudzieho kľúča do iných tabuliek pridať napr. používateľovi jeho adresu.

Pre väčšiu bezpečnosť a zároveň lepší užívateľský komfort sme pridali do nášho návrhu ďalšie dve tabuľky: *Device*, a *Auto\_login*, ktoré majú za úlohu uložiť „device id“ a zistiť, či na danom zariadení bol už užívateľ niekedy prihlásený. Pokiaľ existuje záznam o takom prihlásení, systém užívateľa automaticky prihlási. Táto funkcia je pre PHP nie úplne ideálna a potrebná, no v prípade prechodu „frontendu“ na iný programovací jazyk poprípade vytvorenia mobilnej aplikácie, ktorá bude pristupovať k rovnakému backendu, bude táto funkcia veľmi užitočná a pre používateľov príjemná.

### Entitno-relačný diagram

Entitno-relačný diagram zobrazuje štruktúru databázy v grafickej podobe, kde obdĺžniky predstavujú jednotlivé tabuľky, elipsy znázorňujú atribúty danej tabuľky a kosoštvorce zase vzťahy medzi tabuľkami databázy.

Ako môžeme vidieť na obr. 2.1, náš návrh pozostáva zo šiestich „hlavných“ tabuliek a jednej prepojovacej tabuľky:

* Address, s atribútmi adresy:
  + state, ako údaj o štáte
  + region, ako údaj o regióne
  + city, ako údaj o meste resp. obci
  + street, ako údaj o názve ulici
  + street\_number, ako údaj o čísle domu
  + postal\_code, ako PSČ (poštové smerovacie číslo)
* User, s atribútmi používateľa:
  + fname, ako meno
  + lname, ako priezvisko
  + email, ako emailová adresa a prihlasovacie meno
  + phone, ako telefón
  + description, ako nejaký dobrovoľný popis užívateľa
  + role, ako používateľová rola (prístupové práva)
  + tag, ako nejaké označenie (status) daného používateľa, (napr. deaktivácia účtu)
  + created\_DT, ako dátum a čas vytvorenia účtu
  + last\_signin, ako dátum a čas posledného prihlásenia,
  + password, ako prihlasovacie heslo
  + is\_verified, ako booleovská hodnota, či bol daný užívateľ overený (napr. emailom). Táto funkcia je momentálne len príprava na budúce funkcie. Overenie emailom momentálne nie je funkčné z dôvodu, že na posielanie emailov je potrebné mať stránku na hostingu, kde je dostupný emailový poštový server.
* Company, s atribútmi firiem:
  + title, ako názov firmy
  + description, ako popis firmy, najčastejšie využívaný ako údaj o zápise do živnostenského registra
  + email, ako kontaktný email firmy
  + phone, ako telefonický kontakt
  + ico, ako IČO firmy
  + dic, ako DIČ firmy
  + icdph, ako IČ DPH, pokiaľ je táto hodnotá *null* jedná sa o podnikateľa, ktorý nie je platcom DPH
  + iban, ako bankové spojenie prostredníctvom IBAN
  + swift, ako SWIFT kód banky
  + bank, ako názov banky
* Invoice, a atribútmi faktúry:
  + title, ako názov resp. číslo faktúry
  + description, ako dodatočný popis faktúry
  + total, ako suma bez DPH
  + vat, ako vyrátaná suma DPH
  + total\_vat, ako suma s DPH
  + status, ako status faktúry napr. 2 – zaplatená
  + created, ako dátum vytvorenia faktúry
  + suplied, ako dátum dodania
  + due\_date, ako dátum splatnosti faktúry
* Invoice\_item, s atribútmi faktúrovanej položky:
  + title, ako názov položky
  + description, ako voliteľný popis položky
  + quantity, ako množstvo danej položky na faktúre
  + unit, ako merná jednotka danej položky
  + price, ako jednotková cena (za jednu mernú jednotku)
* Device, s atribútmi zariadenia pristupujúceho na systém:
  + created\_at, ako časová pečiatka kedy bol daný záznam vytvorený
  + last\_login, ako časová pečiatka posledného prístupu
* Auto\_login, ktorá nemá vlastné atribúty okrem primárneho a cudzích kľúčov. Tie si rozoberieme v odstavci nižšie.

Obrázok, na ktorom je kresba, vzor, náčrt, výšivka

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

Obr. . Entitno-relačný diagram

Každá z tabuliek potrebuje jedinečný identifikátor, tzv. primárny kľúč, ktorý jednoznačne určuje o aký záznam ide. Tabuľky *User* a *Company* budú takisto obsahovať cudzí kľúč, ktorý bude odkazovať na tabuľku *Address*, z toho dôvodu, aby bolo možné priradiť samostatnú adresu firme aj používateľovi - predpoklad je, že jeden používateľ môže mať aj viacero firiem. Z toho dôvodu je potrebné priradiť firmu k používateľovi a teda tabuľka *Company* obsahuje ďalší cudzí kľúč, ktorý odkazuje na tabuľku *User*.

Tabuľka Invoice má dva cudzie kľúče pritom obidva odkazujú na tabuľku *Company*. Jeden ako dodávateľ (*suplier*) a druhý ako odoberateľ (*customer*). K tabuľke *Invoice\_item* je potrebné priradiť tabuľku *Invoice*, a to preto, aby bolo možné priradiť fakturovanú položku k faktúre.

Pri prihlasovaní používateľa máme pripravenú funkciu automatického prihlásenia, ktorá robí to, že pokiaľ sa používateľ na danom zariadení neodhlásil, tak pri opätovnej návšteve stránky bude automaticky prihlásený. K tomu potrebujeme mať uložený nejaký jedinečný identifikátor zariadenia a spárovať ho s používateľom. Keďže tieto páry sú kombinácie dvoch tabuliek, *User* a *Device*, potrebujeme jednu spájaciu tabuľku – *Auto\_login*. Tá obsahuje dva cudzie kľúče, jeden odkazujúci na tabuľku *Device* a druhý na tabuľku *User*.

Pokiaľ máme takto navrhnutý entitno-relačný diagram, môžme pristúpiť k vytvoreniu databázy spolu s tabúľkami. Tú môžeme vytvoriť napr. SQL príkazmi „CREATE DATABASE ...“ a „CREATE TABLE ...“. Presný kód nájdeme v zdrojových kódoch (bližšie popísané v prílohe).

### Relačný dátový model

Relačný dátový model (RDM) je iným zobrazením štruktúry databázy – návrhu tabuliek a vzťahov medzi nimi. RDM si môžeme nakresliť samy ale väčšina databáz ponúka možnosť vykresliť RDM automaticky na základe vytvorených tabuliek a vzťahov.

Obrázok, na ktorom je text, snímka obrazovky, softvér, webová stránka

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

Obr. . Relačný dátový model v rozhraní phpmyadmin s označenými vzťahmi

Na obr. 2.2 je zobrazený RDM diagram vykreslený v phpmyadmin (webovom rozhraní pre prácu s MySQL databázou). Každá karta zobrazuje jednu tabuľku v databáze. Položky zapísané v karte sú atribúty danej tabuľky. Ikona kľúča pri atribúte signalizuje, že daný atribút je primárnym kľúčom danej tabuľky. Prepojenia medzi tabuľkami zobrazujú vzťahy medzi tabuľkami pričom čísla pri prepojeniach signalizujú typ priradenia – napr. prepojenie tabuliek *Invoice* a *Invoice\_item* v Rozvrhu by sa dalo interpretovať takto: „Jednej faktúre môže byť priradených N fakturačných položiek (znak N), pričom jednej fakturačnej položke môže byť priradená len jedna faktúra (znak 1)“. Podobne môžeme interpretovať všetky ostatné vzťahy.

## Návrh systému

Kvôli jednoduchšej škálovateľnosti a prehľadnosti sme systém navrhli z dvoch častí: zo systému, ktorý obsluhuje databázu (backend) a časti systému, ktorá interaguje s užívateľom (frontend). Medzi nimi je dohodnutá striktná štruktúra výmeny dát, principiálne zobrazená na obr. 2.3.

Pokiaľ dôjde k nejakej akcii na frontende, napr. k odoslaniu formulára alebo k načítaniu stránky, kde sú potrebné dáta z databázy, frontend pošle požiadavku (request) na backend a čaká od neho na odpoveď (response). Backend túto požiadavku spracuje (kto ju posiela, čo potrebuje, a aké dáta vyžaduje), zvaliduje resp. overí prístupy a následne vráti odpoveď (status odpovede, či nenastala chyba, správu ak chyba nastala a dáta, pokiaľ chyba nenastala).

Presná štruktúra požiadavky:

$params = array( // request structure

"auth" => array(

"email" => "", // user email that is logged in

"device" => "" // device id

),

"action" => "", // request on backend

"data" => array(

// data that backend needs to return correct data

)

);

$params = json\_encode($params);

$request = array("param" => $params); // request that is being send via POST method to server.php

Presná štruktúra odpovede:

$response = array( // response structure

"status" => "success",

"message" => "",

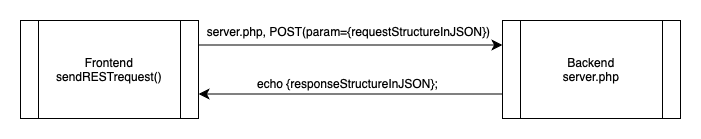
"data" => array(

// data to return

)

);

echo json\_encode($response);



Obr. . Principiálne znázornenie komunikácie medzi frontendom a backendom

### Časť spolupracujúca s databázou – backend

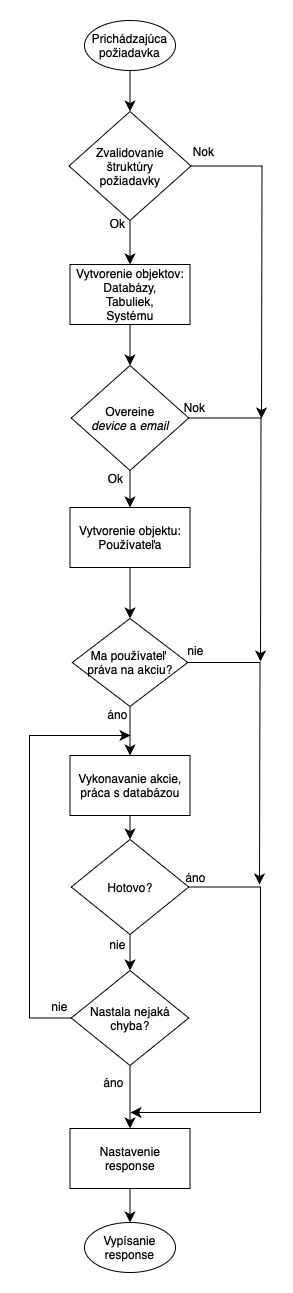
Táto časť systému pozostáva z niekoľkých súborov, ktoré by sme pre väčší prehľad mohli rozdeliť takto:

* Základné súbory:
  + **server.php**, hlavný – ten, ktorý voláme pri požiadavke
  + globals.php, obsahuje všetky potrebné inicializačné a filtračné dáta
  + functions.php, obsahuje základné funkcie systému, napr. *filter\_data()*...
* Súbory, v ktorých sa nachádzajú zadefinované triedy (classes):
  + classSystem.php, obsahuje metódy, ktoré priraďujú (zisťujú) práva používateľa, ktorý poslal požiadavku na dáta
  + classDb.php, obsahuje atribúty a metódy potrebné pre prácu s databázou (connect, build query, ...)
  + classTable.php, obsahuje atribúty a metódy potrebné pre prácu s tabuľkou (select, insert, update, delete...)
  + classUser.php, obsahuje triedy odstupňované podľa role používateľa (guest, user, admin, ...). Tie následne obsahujú metódy, ktoré môže vykonávať používateľ s danou rolou. Tieto metódy sú kvôli prehľadnosti rozdelené do samostatných súborov pomocou *trait* objektu v PHP.
* Trait objekty obsahujúce metódy, ktoré priamo interagujú s jednotlivými tabuľkami databázy (*db\_tb*):
  + db\_address.php, obsahuje metódy, pracujúce s tabuľkou *Address*
  + db\_auto\_login.php, obsahuje metódy, pracujúce s tabuľkou *Auto\_login*
  + db\_company.php, obsahuje metódy, pracujúce s tabuľkou *Company*
  + db\_device.php, obsahuje metódy, pracujúce s tabuľkou *Device*
  + db\_invoice\_item.php, obsahuje metódy, pracujúce s tabuľkou *Invoice\_item*
  + db\_invoice.php, obsahuje metódy, pracujúce s tabuľkou *Invoice*
  + db\_user.php, obsahuje metódy, pracujúce s tabuľkou *User*
* Trait objekty obsahujúce metódy, ktoré sú komplexnejšie (features – *feat*):
  + feat\_system.php, obsahuje metódy, ktoré sa týkajú systému, ako napr. login, signup, logout, a pod.
  + feat\_invoice.php, obsahuje metódy, ktoré sa viažu k vytváraniu, upravovaniu a správe faktúr, čo zahŕňa nie len vytvorenie záznamu do tabuľky *Invoice* ale aj potrebné úpravy v tabuľkách *Company* a *Invoice\_item*.

Celá táto backendová časť systému sa vykonáva len pri zavolaní súboru server.php a to len s poslanými parametrami pomocou POST metódy, inak táto časť skončí chybovou hláškou. Poslané parametre musia dodržať definovanú štruktúru, ktorú sme si podrobnejšie rozobrali vyššie. Údaje zadané v tejto štruktúre sa navyše kontrolujú a filtrujú podľa vopred definovaných parametrov.

Podľa dát zadaných v súbore globals.php sa vytvorí objekt databázy a postupne sa do neho pridajú objekty jednotlivých tabuliek. Táto štruktúra databázy spolu s tabuľkami musí byť explicitne zadefinovaná v súbore globals.php a musí zodpovedať skutočnej štruktúre databázy.

Následne sa vytvorí objekt triedy System, ktorý podľa priloženého emailu a device v štruktúre requestu, zdeteguje používateľa a jeho rolu. Podľa toho vytvorí objekt triedy Guest, User alebo Admin (poprípade je možné doprogramovať akúkoľvek ďalšiu triedu používateľa), pričom Guest je neprihlásený používateľ, User je používateľ prihlásený a Admin je používateľ prihlásený no s najvyššími právami.



Obr. . Principiálna schéma práce backendu

Následne sa volá požadovaná akcia ako metóda objektu triedy používateľa. Vo všeobecnosti tieto metódy pozostávajú z jednoduchej validácie poslaných údajov z requestu a následnej práce s tabuľkami. Tú sme vyriešili tak, že v kóde stačí zadať aké údaje požadujeme z tabuľky alebo celej databázy a pokiaľ dopyt na databázu nie je príliš komplexný, zavolaním metódy tabuľky napríklad ($User->select($data, $where)) dostaneme ako návratovú hodnotu výsledok dopytu vo forme asociatívneho poľa. Tieto metódy sú 4 základné pre každú tabuľku, a to select, update, insert a delete. Fungujú tak, že ako vstup berú data, ktoré chcete získať alebo vložiť a následne berú where struktúru, pomocou ktorej vieme definovať presne aké dáta. Pričom select zvládne aj spájanie tabuliek teda joining. No pokiaľ chceme vytvoriť omnoho komplexnejšie dopyty na databázu, tak máme k dispozícii metódu objektu triedy databázy, ktorá berie ako vstup SQL dopyt a vráti opäť asociatívne pole výsledkov z daného dopytu. Vo väčšine prípadov si vystačíme s metódami tabuľky, ktoré vytvoria SQL dopyt automaticky, no v niektorých špecifických prípadoch musíme pristúpiť k manuálnemu zadaniu SQL dopytu.

Pokiaľ sme v našej požadovanej akcii (request od frontendu) vytvorili a získali všetky potrebné dáta, tak vytvoríme štruktúru response do ktorej vložíme údaje a pošleme ich naspäť tak, že ich vypíšeme ako JSON string pomocou funkcie echo.

Principiálnu schému pracovania backendu môžeme vidieť na obr. 2.4.

### Časť interagujúca s užívateľom – frontend

Táto časť pozostáva prevažne zo stránok, na ktorých užívatelia interagujú so systémom, takisto zo základných funkcií používaných v tejto časti a zo šablón faktúr (momentálne len jedna, no v budúcnosti je možné ich doplniť):

* Stránky:
  + **index.php**, hlavná stránka, pre prihláseného používateľa zobrazuje jeho štatistiku,
  + p\_admin.php, stránka pre admina, umožňuje priamo vytvárať requesty na backend a testovať jeho funkčnosť,
  + p\_invoice.php, stránka pre vytváranie a editáciu faktúr,
  + p\_lists.php, stránka so zoznamom všetkých faktúr a zákazníkov resp. odoberateľov,
  + p\_login.php, stránka pre prihlásenie sa do systému,
  + p\_logout.php, stránka pre odhlásenie sa zo systému,
  + p\_profile.php, stránka pre editáciu používateľových údajov a údajov jeho firmy
  + p\_signup.php, stránka pre zaregistrovanie sa do systému,
* Components:
  + init.php, základný komponent každej stránky, ktorý ma za úlohu overiť prihlásenie užívateľa pomocou email a device uložených v SESSION a pomocou backendovej akcie *autologin*.
  + settings.php, obsahuje dáta o ceste k priečinku, kde sa nachádza zdrojový kód systému (stránky)
* Functions:
  + QRGenerate.php, obsahuje funkcie pre vygenerovanie QR kódu na faktúru (PayBySqare),
  + BackendConnect.php, obsahuje funkciu, ktorá vytvára request na backend a spracováva result z neho, ktorý vracia ako výsledok z funkcie,
* Invoice\_templates:
  + template3.php, aktuálna funkčná šablóna pre faktúru, pre demonštráciu nie je potrebných viac šablón, no do budúcna je možné ich pridať,
* Ostatné súbory:
  + toPdf.php, stránka zodpovedá za export vyplnenej šablóny faktúry do PDF
  + BackendForm.php, testovacia stránka pre programátora na testovanie dopytov na backend (testovanie backendu)

Pri navštívení akejkoľvek stránky sa najprv zavolá súbor init.php, ktorý zistí, či je v SESSION uložené device\_id. Ak uložené nie je vyžiada sa vygenerovanie nového na strane backendu. Pokiaľ máme uložené nové device\_id, tento komponent ďalej zisťuje, či máme uložený email používateľa, čo indikuje jeho prihlásenie. V prípade, že áno, frontend pošle request na backend o overenie prihlásenia daného emailu na danom device\_id. Ak je overenie správne, používateľ bude prihlásení, ak nie, tak používateľa odhlásime (vymažeme zo SESSION).

Ďalej kód pokračuje renderovaním stránky. Vo všeobecnosti, stránky možu mať svoje podstránky, medzi ktorými sa prepíname pomocou atribútu „page“ v URL query. Ten detegujeme v PHP pomocou metódy GET a podľa toho zobrazujeme obsah na danej stránke. V prípade, že daná stránka alebo podstránka vyžaduje nejaké údaje, pýtame si ich príslušným requestom od backendu.

V prípade, že dôjde k odoslaniu formulára užívateľom, zadané hodnoty sa kontrolujú vlastnými funkciami pre spracovanie zadaných údajov. Ak je v zadaných údajoch nejaká chyba, vypíšeme na koniec formulára hlášku. Ak je formulár vyplnený správne, údaje sú odoslané na backend s príslušným requestom, ktorý ich opäť zvaliduje voči špeciálnym znakom a cross site scriptingu. Následne backend vykoná akciu a vráti status *success* alebo *error*. Podľa toho sa na frontende rozhodujeme, čo robíme. V prípade error-u vypíšeme chybovú hlášku ktorá sa nachádza v *message* atribúde štruktúry odpovede z backendu alebo v prípade statusu *success* pokračujeme v požadovanom procese ďalej.

Principiálna schéma fungovania frontend-u je znázornená na obr.

Obrázok, na ktorom je náčrt, diagram

Obsah vygenerovaný umelou inteligenciou môže byť nesprávny.

Obr. . Principiálne zobrazenie procesu na strane frontendu

Záver

Pre otestovanie nášho systému sme pripravili testovacie dáta, ktoré môžeme spustením SQL skriptu vložiť do databázy a následne overiť funkčnosť systému. Na stránke index.php nás ako nezaregistrovaného používateľa, čaká predstavenie možností systému ako takého s výzvou k registrácií. Môžeme sa zaregistrovať alebo využiť už jeden z predpripravených testovacích účtov a jednoducho sa prihlásiť kliknutím na dané tlačidlo. Po prihlásení nás na hlavnej stránke čaká štatisktika našich vytvorených faktúr, zákazníkov a príjmov. Prekliknutím na stránku profile, môžeme editovať svoje súkromné ako aj firemné údaje. Po kliknutí na stránku Lists môžeme vidieť zoznamy už vytvorených faktúr a ich statusov. Po prekliknutí sa na podstránku customers vidíme zoznam našich zákazníkov, ktorým sme už fakturovali. Ak chceme danému zákazníkovi vytvoriť faktúru znovu, stačí kliknuť na tlačidlo Create Inv. v jeho riadku. Touto akciou sa dostaneme na stránku Add Invoice, na ktorej môžeme vytvárať a spravovať faktúry.

Vytváranie faktúry pozostáva z niekoľkých podstránok logicky rozdelených: základne info na faktúre, dodávateľ, odoberateľ a položky faktúry. Po vyplnení údajov na kazdej podstránke môžme pristúpiť k vytvoreniu faktúry. Ak všetko prebehlo vporiadku, máme vytvorenú faktúru, ktorú následne možeme kliknutím na tlačidlo „Show invoice“ vytlačiť do PDF. V prípade, že potrebujeme faktúru upraviť, jednoducho zmeníme údaje v danej kolonke, uložíme na danej podstránke a stlačíme „Update invoice“. Vytvorená nová faktúra sa zobrazí v zozname faktúr. V prípade, že máme už vytvorenú faktúru a chceme ju len kopírovať, danú faktúru nalistujeme v zozname faktúr, otvoríme a dáme „Copy Invoice“. To nám vytvorí novú faktúru so skopírovanými údajmi z predošlej faktúry. Údaje sa kopíruju všetky okrem položiek, názvu faktúry a dátumov.

V prípade, že prihlásený používateľ má rolu admina, na stránke jeho profilu vidíme odkaz na „admin page“. Po kliknutí na tento odkaz dostávame možnosť posielať „Raw“ dáta (štruktúru) na stranu backendu a tým testovať jeho funkčnosť. Po odoslaní daného requestu vidíme „Raw“ data z backendu zobrazené pod formulárom.

Vykonaním tohto testu na danom systéme vieme povedať, že systém je funkčný. Pre zverejnenie daného systému by bolo potrebné dorobiť zopár riešení ako možnosť zmeniť si heslo, funkciu zabudnutého hesla, overenie emailovej adresy cez poslaný mail, otestovať bezpečnosť a integritu databázy voči útokom, a pod. No daný systém je navrhnutý tak, aby bol ľahko škálovateľný a prehľadný.

**Prílohy**

Zoznam príloh

Príloha A | Návod na spustenie 18

1. Návod na spustenie

Pre spustenie nášho kódu je potrebné mať nainštalovaný XAMPP spolu s MySQL databázou. Následne je potrebné stiahnuť do priečinka htdocs zdrojový kód z GitHub-u (url: <https://github.com/leskopet/AppInvoice>). V phpmyadmin naimportujte súbor Backend/SQL/db\_structure.sql alebo iným spôsobom spuste daný súbor na vašom databázovom serveri. Ako nepovinný bod je možné si v phpmyadmin vytvoriť používateľa pre danú databázu. Následne je potrebné v súbore Backend/globals.php skontrolovať údaje o databáze (host, port, dbname, username, password). Potrebné je skontrolovať takisto v súbore components/settings.php údaje o použitom protokole, hoste (server resp. doména), porte a ceste do priečinka. V prípade, že všetky údaje sedia systém by mal byť plne funkčný.

V prípade, že chceme otestovať daný systém na testovacích dátach v databáze je potrebné spustiť SQL súbor (opäť naimportovaním do phpmyadmin alebo iným spôsobom spustiť) Backend/SQL/test\_data.sql. Týmto krokom by mala byť databáza naplnená testovacími údajmi.

Pre prihlásenie sa ako administrátor môžeme použiť nasledovné údaje:

* Username: [admin@appinvoice.sk](mailto:admin@appinvoice.sk)
* Password: admin

Pre prihlásenie sa ako bežný používateľ môžeme použiť nasledovné údaje:

* Username: [pouzivatel@appinvoice.sk](mailto:pouzivatel@appinvoice.sk)
* Password: heslo