Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Nadižar

Vzpostavitev informacijskega okolja za vodenje evidence bioloških vzorcev

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN MATEMATIKA

doc. dr. Miha Moškon
MENTOR

Ljubljana, 2018

Tematika naloge:

Kandidat naj v svojem delu predlaga informacijsko rešitev za vodenje evidence bioloških vzorcev. Rešitev naj temelji na centralizirani podatkovni bazi, do katere naj uporabniki dostopajo preko spletne aplikacije. Implementacijo rešitve naj demonstrira pri vzpostavitvi informacijskega sistema za izbran laboratorij na Medicinski fakulteti Univerze v Ljubljani.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom doc. dr. Mihe Moškona,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu prek univerzitetnega spletnega arhiva.

— Nejc Nadižar, Ljubljana, marec 2018.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Nadižar

Vzpostavitev informacijskega okolja za vodenje evidence bioloških vzorcev

POVZETEK

Diplomska naloga se ukvarja s problematiko shranjevanja, obdelavo in analizo večjih količin podatkov, ki so rezultat delovnega procesa raziskovalnih laboratorijev. Razvit in implementiran je bil podporni informacijski sistem, ki naslavlja omenjeno problematiko. Tiskano obliko shranjevanja podatkov, ki je bila v ciljnem laboratoriju uporabljena prej, je z vpeljano rešitvijo nadomestila centralizirana podatkovna baza. Interakcija uporabnika s podatkovno bazo poteka preko grafičnega spletnega vmesnika, ki omogoča shranjevanje in prikazovanje podatkov. Informacijski sistem poleg elektronskega evidentiranja vzorcev in shranjevanja podatkov o vzorcih vpelje tudi standardizirano označevanje vzorcev s hitro berljivimi kodami QR, ki predstavljajo unikatno označbo vzorca. Vpelje tudi strukturo fizičnih označb na samem vzorcu, ki mora poleg optične kode vsebovati tudi podatke, nujne za hitro razpoznavanje vzorca.

V delu se seznanimo z uporabljenimi tehnologijami in metodologijami. Pregledamo celoten postopek implementacije in razvoja posameznih delov sistema. Na koncu opišemo, in sicer korake procesa, v katerega je vključen posamezen vzorec v ciljnem laboratoriju. Od označevanja in evidentiranja izvornega vzorca do zaključka eksperimenta z izdelavo poročila o analizi pridobljenih podatkov za posamezen vzorec.

Ključne besede: informacijski sistem, kodiranje QR, evidentiranje vzorcev, funkcijska genomika

University of Ljubljana
Faculty of Computer and Information Science

Nejc Nadižar

Information system for management of biological samples

ABSTRACT

The thesis addresses the problematic of handling, storage and analysis of large quantities of data, which are the product of the working process in research laboratories. A supporting information system that addresses the aforementioned issue was developed and implemented. Printed data records were replaced with centralized database environment. User interaction with the database was realized via a graphic user interface, which provides input and display of data. In addition to the electronic recording of samples and storage of sample data, information system also introduces standardized markings of samples with easily readable QR codes, which provide unique sample identifiers. The system also introduces the structure of physical labels on the sample itself, which, in addition to optical code, must also contain the data necessary for rapid recognition of samples.

We get acquainted with the technologies and methodologies used. We review the entire process of development and implementation of individual components of the system. Finally, we describe the steps of the process in which samples are included, from labeling and recording the initial sample, to the conclusion of the experiment with the report of the analysis of obtained data for each sample.

Key words: information system, QR code, sample documentation, functional genomics

ZAHVALA

Zahvaljujem se mentorju doc. dr. Mihi Moškonu za vso potrpežljivost in nasvete pri pisanju diplomskega dela. Prav tako se zahvaljujem tudi družini, prijateljem, sodelavcem in vsem, ki so mi v času študija nudili neizmerno oporo in pomoč.

— Nejc Nadižar, Ljubljana, marec 2018.

KAZALO

1 Uvod

Trenutno se nahajamo v sodobni informacijski dobi, kjer imamo z vsakim našim dejanjem možnost ustvariti ogromne količine podatkov. S količino podatkov se povečuje tudi potreba po podpornih sistemih za ravnanje s podatki. Ravnanje s podatki vsebuje vse od enostavnega shranjevanja podatkov, pa vse do kompleksnejših obdelav podatkov. Poleg samega shranjevanja in obdelave potrebujemo tudi kakovost, dostopnost in varnost le-teh. Kakovost lahko razdelimo na več lastnosti podatkov. Enak tip podatka mora biti zapisan v istem formatu zapisa. Pravilnost in konsistenstnost se ukvarjata s vrednostmi samih podatkov, v primeru, da se podatek nahaja na več različnih mestih ima povsod enako vrednost. Dostopnost lahko razumemo kot možnost dostopa do vseh podatkov, katere v danem trenutku potrebujemo. Varnost pa povezuje kakovost in dostopnost, saj si z njo zagotovimo dostop do podatkov, vnos in spreminjanje le-teh samo pooblaščenim uporabnikom. Sistemi za ravnanje s podatki le redko dohajajo sisteme s katerimi le-te ustvarjamo. Shranjevanje podatkov o raziskavah, eksperimentih v preglednicah, tekstovnih datotekah lokalno pri vsakem uporabniku lahko zgleda smiselno za manjšo organizacijo. Vendar pa je v primeru širitve organizacijo smiselno razmišljati o podpornem informacijskem sistemu. Informacijski sistem nam zagotavlja centralizirano shranjevanje podatkov, s katerim si zagoto-

vimo kvaliteto podatkov, saj je vsak podatek shranjen enkrat, le prikažemo ga večkrat. Zaradi centraliziranosti imamo tudi eno samo dostopno točko do podatkov, ki jo moramo zavarovati, s čim povečamo varnost podatkov. Poleg vseh naštetih kvalitet elektronskega informacijskega sistema, pa je prednost tega kot medij shranjevanja podatkov tudi v fizični velikosti prostora, ki ga shranjeni podatki zasedajo.

V okviru diplomske naloge smo razvili in implementirali podporni informacijski sistem za vodenje evidence bioloških vzorcev. Potreba po takem sistemu se je pokazala že v samem začetku dela z vzorci, saj količina zajetih vzorcev in podatkov z njimi povezanih hitro postane neobvladljiva za ročno vodenje. Sistem obsega tako evidentiranje novih vzorcev in shranjevanje podatkov o vzorcu, ki smo jih pridobili čez celoten potek raziskave. Tako si zagotovimo sledenje, kaj se je z vzorcem dogajalo ter tudi celotno zgodovino vzorca. Informacijski sistem se ukvarja tudi s fizičnim označevanjem vzorcev. Vpeljali smo standardizacijo oznak za vzorce, ne glede na vrsto vzorca. Zaradi kompleksnega kodiranja unikatnih identifikatorjev vzorcev, smo vpeljali tudi optični sistem kodiranja za lažjo razpoznavo vzorcev.

Implementacija sistema je vsebovala dodatno konfiguracijo strežnika in namestitev ter konfiguracijo pripadajoče programske opreme, programskega okolja za povezavo z relacijskim okoljem in podatkovno bazo. Programska oprema, ki smo jo uporabili za operacijski sistem osnovnega strežnika, spletni strežnik, programsko in relacijsko okolje je odprtokodna in brezplačno dostopna. Razvili smo svojo programsko opremo za grafični uporabniški vmesnik ter za povezavo med vmesnikom in podatkovno bazo.

S podpornim informacijskim sistemom smo želeli zagotoviti centralizirano shranjevanje in vnos podatkov, hitrejše analize ter obdelavo podatkov. Poleg tega smo želeli zagotoviti tudi varnost, modularnost in skalabilnost sistema. Varnost zaradi same narave podatkov. Modularnost, da bi lahko sistem brez večjih težav priredili drugačnemu tipu in strukturi podatkov, in enostavno skalabilnost v primeru večje količine podatkov ali razširitve sistema. Celoten sistem je bil nameščen in se uporablja na Centru za funkcijsko genomiko in bio-čipe (CFGBC) Medicinske fakultete Univerze v Ljubljani.

2 Izbira tehnologij in metodologij

Evidentiranje predmetov nam omogoča vpogled v njihovo zgodovino ter nam zagotavlja sledljivost skozi celoten postopek obdelave. S pomočjo evidence podatkov lahko za vsako časovno obdobje predmeta določimo, kje se je nahajal in kaj se je z njim dogajalo. Fizično označevanje predmetov pa nam omogoča, da iz večje količine podobnih ali enakih predmetov izberemo točno določenega, ki nas v danem trenutku zanima.

2.1 Označevanje vzorcev

V našem primeru smo bili glede fizičnega označevanja vzorcev precej omejeni zaradi fizične velikosti epruvet v katerih so vzorci shranjeni. Poleg tega so zaradi obstojnosti tkiva vzorci shranjeni pri ekstremnih pogojih, kar je pomenilo, da smo potrebovali namensko opremo.

2.1.1 Nalepke

Shranjevanje vzorcev krvi zaradi obstojnosti le-te poteka pri nizkih temperaturah (–80 °C). Temperaturo je bilo potrebno upoštevati pri izbiri in obstojnosti oznak vzorcev, saj niso vsi tipi nalepk namenjeni enakim pogojem. Potrebovali smo obstojnost za daljše shranjevanje pri nizkih tem-

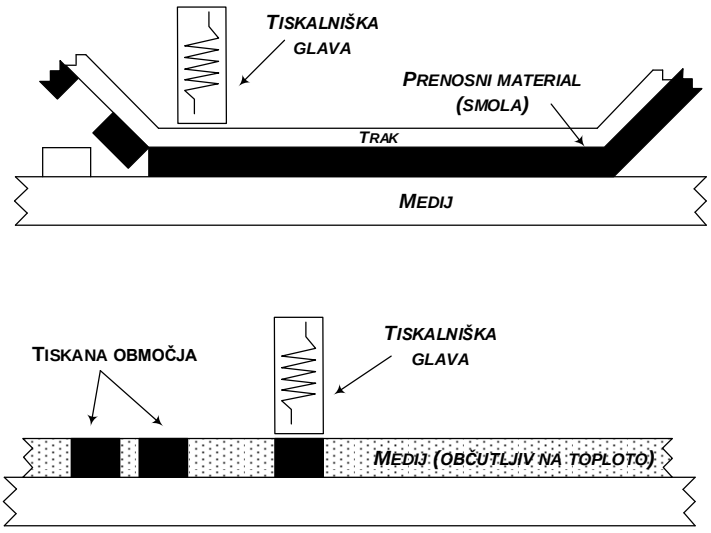
peraturah in pri nadaljnjem zamrzovanju in odmrzovanju. Zaradi slednjega morajo biti odporne proti nabiranju in zmrzovanju kondenza, zaradi katerega bi nalepka lahko odstopila od epruvete.

Pomembna je tudi obstojnost tiska na nalepki, ki je odvisen od samega materiala nalepke, kvalitete in metode tiskanja. Čeprav bi bilo možno doseči skoraj enako kvaliteto tiska na nalepki z navadnim laserskim tiskalnikom, smo se odločili za namenski tiskalnik nalepk, saj ima za ta namen prilagojeno metodo tiskanja in namensko črnilo.

V splošnem sta za namen evidentiranja najbolj razširjeni dve metodi tiska, prva je imenovana *thermal direct*, kar pomeni tiskanje brez uporabe črnila. To pomeni, da že nalepka sama vsebuje črnilo. Toplotni člen tiskalnika, tako samo segreje določene dele nalepke s katerimi aktivira črnilo, da se del pobarva. Prednost takega tiska je, da je cenovno ugoden in brez uporabe črnila. Slabosti pa ima pri obstojnosti, saj je primeren za nalepke, ki so namenjene označevanju za krajša časovna obdobja (obstojnost tiska je okoli enega meseca).

Druga metoda je imenovana *thermal transfer* in je podobna samemu laserskemu tiskanju. Črnilo se prenese na nalepko s pomočjo statike in nato še dodatno utrdi preko toplotnega člena. Prednost takega tiska sta precej daljša obstojnost in kvaliteta za zgolj majhno razliko v stroških [1, 2].

Obe metodi tiskanja sta prikazani na sliki 2.1.



Slika 2.1 Zgornja slika prikazuje metodo *thermal transfer*, spodnja pa *thermal direct*.

Ker sta obstojnost in kvaliteta pomembni, smo se odločili za drugo metodo, saj so izvorni vzorci krvi lahko shranjeni tudi do 6 mesecev, preden se nabere zadostna količina vzorcev za izvedbo njihovih analiz. Dodatni vzorci krvi (okoli 2 ml), ki jih dobimo pri razdelitvi krvi za izolacijo DNA, so shranjeni tudi za precej daljša časovna obdobja.

2.1.2 Optično kodiranje znakov

Kot smo že omenili, smo pri razpoznavanju vzorcev omejeni glede velikosti podatkov, ki jih lahko fizično prikažemo na samem vzorcu. Zato smo se odločili, da unikaten niz vzorca prikažemo z optično predstavitvijo. Slednja lahko vsebuje večjo gostoto znakov kot alfanumerična predstavitev. Poleg tega so optične kode lažje berljive za elektronske naprave z optičnim razpoznavanjem.

Primer optične kode je črtna koda (angl. *barcode*), ki je dandanes ena najbolj razširjenih in s katero se srečujemo dnevno. Zaradi hitrega napredka in informatizacije se je povečala tudi potreba po optičnih kodah, ki bi znale zakodirati na enaki površini večje količine znakov. Zaradi tega so se v industriji razvile dvodimenzionalne optične kode, med katere spada tudi kodiranje QR (angl. *Quick Response Code*). Dvodimenzionalne kode vsebujejo veliko več podatkov v primerjavi z linearnimi. Za primerjavo, črtna koda hrani okoli 30 alfanumeričnih znakov, medtem ko koda QR lahko vsebuje do 7000 poljubnih znakov [3]. Primerjavo med kodo QR in črtno kodo prikazuje slike 2.2.

Kodiranje QR združuje več prednosti predhodnih optičnih kod:

- Branje ne glede na orientacijo kode: zaradi dvodimenzionalnosti kode, je branje neodvisno od orientacije kode.
- Hitro razpoznavanje: ker so točno določeni razpoznavni vzorci sestavni del vsake kode, je branje lahko do 20-krat hitrejše.
- Poljubni znaki: podprtih je več naborov znakov in ti so optimalnejše zakodirani (zavzamejo manj prostora v sami kodi).
- Redundanca: koda ima vgrajeno tudi korekcijo napak (od 7 % do 30 %) v primeru fizične poškodbe kode (npr. odtrgan del ali razmazan tisk).
- Enkripcija: za kriptiranje kode QR potrebujemo samo kodno tabelo.

Slika 2.2 Primerjava kode QR (levo) in črtne kode (desno).

2.2 Evidentiranje vzorcev

Evidentiranje vzorcev predstavlja zapisovanje vseh potrebovanih podatkov v urejen sistem. Zaradi lažjega shranjevanja podatkov, administracije samega sistema in razpoložljive opreme smo se odločili implementirati svoj informacijski sistem. Večino razvoja je bilo izvedeno na programskem delu, saj je bila strojna oprema že zagotovljena.

Programska oprema

Kot osnovo za kakršnokoli upravljanje strojne opreme računalniškega sistema uporabljamo programsko opremo. Glede na samo količino programske opreme s podobno namembnostjo, izbira prave programske opreme ni trivialen postopek [4]. V nadaljevanju je opisan postopek izbire, ki smo ga zaradi velikosti našega problema spremenili, saj nismo potrebovali tako podrobne in kompleksne izbire. Pred samo postavitvijo kriterijev za izbiro moramo določiti:

- naloge, ki jih bo programska oprema opravljala,
- težave, s katerimi se lahko srečamo pri implementaciji nove programske opreme v že obstoječ sistem,
- prednosti, ki jih bo programska oprema prinesla,
- kompromisi, katere bomo morali upoštevati pri implementaciji.

Zgoraj navedene točke nam pomagajo, da že na samem začetku zožimo začetni nabor ustrezne programske opreme. V nasprotnem primeru se lahko zgodi da imamo v naslednjem koraku težjo nalogo izločevanja, saj je treba vsako programsko opremo bolj natančno pregledati. Prilagojeni kriteriji za izbiro primerne programske opreme so vključevali:

- **Lastnosti**: kakšne naloge bo programska oprema opravljala? Vsebuje tudi kakšne druge funkcionalnosti, ki bi nam pomagale pri implementaciji ali povezavi z obstoječim informacijskim sistemom?

- **Kompatibilnost**: kako se programska oprema primerja z ostalo opremo v sistemu? Ali zadostuje standardom, ki jih potrebujemo?

- **Nadaljnji razvoj**: ali bo razvoj programske opreme aktiven? Ali so načrtovane nadgradnje in popravki? Kakšen bo odnos razvijalcev do predhodno razvite programske opreme, če ta obstaja?

- **Prilagodljivost**: kako prilagodljiva bo programska oprema drugim spremembam v sistemu (vpeljava nove programske opreme, nadgradnja informacijskega sistema, zamenjava strojne opreme ipd.)? Kako skalabilna bo v primeru povečanja potreba informacijskega sistema?

- **Stroški**: koliko sredstev bo namenjenih za implementacijo? Ali bomo lahko pokrili stroške licenc, vzdrževanj, nadgradenj?

Strojno opremo na kateri se programska oprema izvaja sestavljajo: fizični strežnik, z operacijskim sistemom Windows 2008 Server R2 in z dvema Intel Xeon E5520 procesorjema, 32 GB delovnega spomina in 8 TB diskovnih enot v RAID1 konfiguraciji, kar pomeni okoli 4 TB razpoložljivega prostora. Na strežniku je nameščeno orodje za virtualizacijo Miscrosoft Hyper-V, preko katerega se izvaja naš ciljni virtualni strežnik. Slednega sestavlja operacijski sistem Scientific Linux release 6.8 (Carbon). Virtualno okolje ima na voljo 16 GB delovnega spomina in 1 TB diskovnega prostora.

Na virtualnem strežniku se poleg naše nameščene programske opreme izvajajo tudi ostala orodja za analizo in shranjevanje podatkov, ki so potrebna za normalno delo laboratorija.

Operacijski sistem

Operacijski sistem Linux (v nadaljevanju Linux) je poleg operacijskega sistema Windows (v nadaljevanju Windows), eden najbolj razširjenih. Medtem, ko je Windows, zaradi svoje enostavnosti uporabe bolj primeren za povprečne uporabnike, je Linux bolj primeren za zahtevnejše uporabnike. Slednjega najdemo v uporabi kot osnovo sistema v raziskovalnih in tehničnih strokah, kjer je pomembna stabilnost, zmogljivost in zanesljivost. V primerjavi z Windows so distribucije

v veliki večini odprtokodne in brezplačne. Obstajajo tudi plačljive distribucije, ki so namenjene večjim podjetjem, kjer pa samemu operacijskemu sistemu pripada tudi podpora.

Na virtualnem strežniku je nameščena distribucija Scientific Linux release 6.8 (Carbon), 64-bitna verzija, z verzijo jedra 2.6.32. Osnova je jedro distribucije Red Hat Enterprise Linux (RHEL), ki je namenjena podjetjem. Distribucija Scientific Linux je bolj primerna za znanstvene namene, ne toliko zaradi orodij vključenih v sam operacijski sistem, ampak zaradi same stabilnosti in obsega sistema. Razvoj Scientific Linux-a poteka v raziskovalnem laboratoriju Fermilab [5].

Spletni strežnik

Ker so računalniki po laboratorijih lahko slabo zmogljivi, stari ali imajo nameščeno starejšo programsko opremo, smo se odločili, da bo vsa interakcija s podatkovno bazo potekala preko spletnega vmesnika, za kar pa na strežniku potrebujemo spletni strežnik. Medtem ko bo odjemalec pošiljal zahteve direktno na strežnik, bo namen strežnika zahteve interpretirati, posredovati relacijskem okolju podatkovne baze, ter odjemalcu vrniti zahtevane podatke ali rezultate. Tako smo se izognili nepotrebnemu delu na samih odjemalcih.

Čeprav obstaja veliko različnih spletnih strežnikov, je eden od najbolj razširjenih Apache HTTP Server (v nadaljevanju Apache). Če bi ga primerjali z bolj komercialnim produktom, kot je na primer Microsoft IIS, je Apache cenejši, ker spada pod odprtokodno licenco. Boljša je tudi dokumentacija ter nabor modulov, s katerimi razpolaga. Izbrali smo ga zaradi same razširjenosti, varnosti ter majhne velikosti same programske opreme. Tako se spletni strežnik izvaja na programski opremi Apache, verzije 2.2.15 [6].

Programski jezik

V današnjem času nam res ni treba skrbeti glede pomanjkanja programskih jezikov, saj se starejši razvijajo naprej, medtem ko skoraj vsakodnevno nastajajo novi. Jeziki segajo od najbolj zmogljivih do najbolj intuitivnih. Večina razvoja v našem času se odvija v okolju spletnih aplikacij. Glede samega nabora programskih jezikov, ki bi bili primerni za razvoj našega sistema, smo morali izbrati takega, ki bo najbolj enostaven, zmogljiv in obsežen.

Za programski jezik samega spletnega vmesnika in povezavo le tega z bazo smo izbrali programsko opremo in programski jezik PHP verzije 5.3.0 [7]. Za to izbiro smo se odločili, ker programska oprema teče zgolj na strežniški strani. Vsi preračuni in prikazovanja se zato izvedejo na strežniku, kar pomeni boljšo odzivnost zaradi boljše zmogljivosti strežnika v primerjavi

z uporabnikovim računalnikom in posledično večja varnost samega sistema.

Podatkovna baza

Podatkovne baze so kot ena od najbolj razširjenih digitalnih oblik shranjevanja podatkov v današnjem času. Trenutno shranjevanje podatkov v ciljnem laboratoriju poteka preko tekstovnih datotek, razpredelnic (Microsoft Office Excel) ter podatkovnih baz Microsoft Access. Slednje so zaradi slabe zmogljivosti (pri velikih bazah) in omejenosti terminalov bolj primerne za lokalno obdelavo podatkov. Prej omenjeni sistemi, ki so trenutno v uporabi, niso namenjeni trajnemu shranjevanju in ažurnosti podatkov. V večini primerov so datoteke pri uporabnikih podvojene. Ker ima vsak uporabnik svojo različico, lahko nekdo po pomoti zamenja novejšo datoteko s staro in tako povozi aktualno stanje ali datoteke enostavno pobriše. Trenutno stanje zaradi decentraliziranosti ni primerno za izvajanje kompleksnejših analiz nad podatki.

Zaradi vseh zgoraj naštetih težav in omejitev, smo se odločili da podatke začnemo hraniti v sistemu, ki omogoča centralizirano shranjevanje večje količine podatkov in pri tem obstaja odziven. Na razpolago je veliko različnih okolij podatkovnih baz [8]. Pri tem delimo podatkovne baze na dva modela:

1. **Dokumentni model**: podatkovna baza je sestavljena iz posameznih dokumentov, kjer vsak hrani svojo različico podatkov. Tukaj lahko pride do problema. Če je enak podatek shranjen v več različnih dokumentih, lahko pride do nekonsistentnosti podatkov, ker se spremembe ne propagirajo čez vse dokumente. Prednost dokumentnega modela je enostavnost vzpostavitve, ter horizontalna skalabilnost.

2. **Relacijski model**: pri tem modelu imamo v celotni podatkovni bazi shranjene zgolj podatke. V istem okolju imamo lahko več različnih podatkovnih baz, katere lahko delimo glede na namembnost. Znotraj vsake podatkovne baze imamo več tabel v katerih so shranjeni prej omenjeni podatki. Za prikaz lahko izberemo podatke, ki ustrezajo določenemu pogoju, ne glede na to ali so podatki razdrobljeni po več tabelah. Ker je podatek v podatkovni bazi shranjen samo enkrat, se ob posodobitvi na kateremkoli mestu, posodobijo vse instance, oziroma kopije tega podatka. Prednost relacijskega modela je ažurnost in konsistentnost podatkov, medtem ko je model slabo horizontalno skalabilen, saj deljenje baz po več različnih strežnikih ni čisto trivialno.

V našem primeru smo izbrali relacijski model, saj imamo normalizirane podatke, kar pomeni, da je potreba po večkratnem hranjenju enakega podatka minimalna. Naši podatki so dobro

strukturirani, za njih pa je značilno, da že vnaprej vemo v kakšni obliki bodo podani ter kakšen bo njihov tip. Na podlagi takih podatkov, lahko zato podatkovno bazo enostavno strukturiramo. Vse stolpce v tabelah in njihove podatkovne tipe lahko vnaprej definiramo. S tem se izognemo vpisu napačno strukturiranih podatkov ter pridobimo na samem prostoru, saj so velikosti podatkov dobro omejene.

Okolje podatkovne baze realizirane na strežniku, je relacijski sistem za upravljanje z bazami MySQL. Verzija programske opreme je 5.1.73 for RHEL [9]. V tem okolju sta postavljeni produkcijska in razvojna podatkovna baza. Okolje MySQL smo izbrali, zaradi njegove razširjenosti in odprtokodne narave. Njegova prednost je majhna velikost, ki jo okolje zaseda na samem datotečnem sistemu. MySQL ima tudi najbolj razširjeno sintakso ter zaradi tega enostaven dostop do pomoči v primeru napak in odpravljanju le-teh.

2.3 Varnost in dostop do podatkov

V današnjem času se vedno več podatkov shranjuje v digitalni obliki, ne samo zaradi enostavnosti shranjevanja, ampak tudi zaradi same fizične velikosti, ki jo take količine podatkov zavzamejo. Ena izmed prednosti je tudi lažji dostop do podatkov. Za dostop ne potrebujemo več fizičnega dostopa do prostora shranjevanja, ampak lahko do podatkov dostopamo tudi oddaljeno. Slednje pomeni, da je dostop nepooblaščenim osebam lahko enostavnejši. Iz tega razloga moramo ključne in zaupne podatke ustrezno zaščititi pred nedovoljenim branjem. Za ta namen obstaja več različnih metod zaščite samih podatkov ali omejitev dostopa do podatkov.

- **Kriptiranje**: metoda podatke kriptira na podlagi računalniškega algoritma s podanim ključem ali geslom, s katerim naredi translacijo znakov. Poznamo več različnih algoritmov. Osnovni niz ali sporočilo imenujemo čistopis, šifrirano pa tajnopis (kriptogram). S tem sicer ne preprečimo dostopa do podatkov, vendar bo uporabnik brez ključa dobil le tajnopis, ki ni nikakor nujno podoben originalu, ne po vsebovanih znakih niti po dolžini.
- **Maskiranje**: pri tej metodi, izvornih podatkov ne spreminjamo, jih samo drugače naredimo neberljive. Pomeni, da jih lahko vizualno popačimo, podatka sploh ne prikažemo, ali prikažemo samo del podatka ter ostalo nadomestimo s poljubnimi znaki. Medtem ko kriptiranje v veliki večini pretvori izvorne podatke na izvorni lokaciji, jih maskiranje pretvori zgolj za namene prikaza.
- **Avtorizacija**: če smo v prejšnjih dveh točkah govorili o pretvorbi podatkov samih, se tukaj ukvarjamo z dostopom do podatkov ali sistema samega. Medtem, ko lahko ločimo

fizično in programsko avtorizacijo, prva za nas ni bila pretirano zanimiva, ker je že urejena z omejitvijo dostopa do fizične lokacije. S programsko avtorizacijo je preprečen dostop do podatkov ali sistema tistim uporabnikom, ki za dostop niso avtorizirani. To pomeni, da nimajo veljavnega uporabniškega imena in gesla ali kakšne druge metode identifikacije, ki bi jim omogočila dostop.

Pri medicinskih podatkih je lahko varnost in anonimnost še posebej pomembna, saj gre za zelo osebne podatke, ki so enostavno izkoriščeni z namenom oškodovanja osebe [10]. Zaradi zagotavljanja anonimnosti smo morali poskrbeti, da na podlagi naše evidence, ni mogoče dostopati do osebnih podatkov pacienta. To smo dosegli tako, da smo za potrebe laboratorija na nevrološki kliniki kreirali nalepke sestavljene iz 4-mestne številke in kode QR, v kateri je zakodirana prej omenjena številka (zgolj zaradi lažjega vpisovanja s čitalcem kod QR). Številke so bile generirane v naključnem vrstnem redu, zato je možnost, da bi zaporedna številka pomenila isti vzorec kot je zapisan v podatkovni bazi majhna. Zaradi tega, je možno osebne podatke povezati z rezultati zgolj z našo kodo QR, kodo z zaporedno številko in evidenco nevrološke klinike.

Poleg same anonimnosti podatkov, je bilo potrebno tudi omejiti dostop do podatkovne baze. Ker podatkovna baza na začetku ni bila namenjena za širšo oziroma splošno uporabo, je bil dostop do spletnega vmesnika in podatkovne baze omejen in mogoč zgolj znotraj lokalnega omrežja samega laboratorija. Za uporabo spletnega vmesnika se mora uporabnik prijaviti z uporabniškim imenom in geslom. Registracija uporabnikov poteka ročno. Administrator ustvari uporabnike in njihov gesla, katere pa morajo nujno spremeniti ob prvi prijavi.

3 Implementacija

Implementacijo si ponavadi predstavljamo kot preslikavo same ideje v realno okolje. V tem koraku se pokažejo vse pomanjkljivosti naše ideje in implementacije le-te ter podrobnosti, na katere nismo pomislili ali pa jim nismo posvetili dovolj pozornosti. V večini primerov napak bomo morali popraviti osnovno idejo in jo potem ponovno prenesti v dejanski sistem. Ta postopek ponavljamo, dokler ideje ne prenesemo v celoti brez modifikacij. V realnem svetu za takšen postopek in tolikšne ponovitve največkrat nimamo ne časa niti sredstev. Velikokrat se odločimo za neko srednjo pot, kar pomeni, da v ideji odpravimo večje pomanjkljivosti, manjše popravke pa implementiramo med samo namestitvijo ali delovanjem, da se čim bolj približamo začetni ideji implementacije.

3.1 Fizično označevanje vzorcev

Vsak vzorec je fizično označen s svojo unikatno nalepko, ki je sestavljena iz dveh delov. Velikost in struktura nalepk je enaka za tri različne vzorcev. Zaradi lažje predstave sta različna dela nalepk poimenovana kot tekstovni in kodni del nalepke.

Nalepke so velikosti 30x15 mm in so primerne za nizke temperature do –80 °C.

3.1.1 Tekstovni del nalepke

Tekstovni del nalepke je sestavljen iz več nizov alfanumeričnih znakov. Število nizov je odvisno od tega za kateri vzorec gre. Čez celoten proces evidentiranja in testiranja smo se srečali s tremi različnimi tipi vzorcev in s tremi različnimi tipi nalepk. Različni tipi nalepk so prikazani na sliki 3.1.

- 10 ml vzorec krvi: vzorec krvi je odvzet pacientu direktno na nevrološki kliniki, preden ga pooblaščena oseba prinese v laboratorij. Tekstovni del nalepke tega vzorca je sestavljen iz dveh vrstic. Prva vrstica vsebuje niz treh alfanumeričnih znakov. Prvi znak nam pove kodo inštituta, naslednja dva pa kodo projekta, kateremu pripada vzorec. V drugi vrstici se nahaja numeričen znak, ki nam pove zaporedno številko vzorca.

- 1,5 ml vzorec DNA: vzorec že izolirane DNA, pripravljen na nadaljnjo obdelavo. Pri tem vzorcu je tekstovni del sestavljen iz treh vrstic. V prvi vrstici imamo zopet tri znake, ki nam opisujejo inštitut in projekt. V drugi vrstici je zaporedna številka, tokrat z dodatkom poševnice (angl. *slash*) in enega numeričnega znaka, iz katerega razberemo, da gre za vzorec DNA. Zadnjo vrstico sestavljajo oznaka škatle, v kateri se shranjujejo vzorci, ter oznaka pozicije v škatli (8 znakov ločenih s poševnico).

- 2 ml vzorec krvi: vzorec krvi kot rezerva ali za uporabo v kakšni drugi raziskavi. Nalepka je podobna kot pri zgornjem vzorcu DNA. Razlikujeta se samo v zadnjem znaku v drugi vrstici, iz katerega lahko razberemo, da gre za vzorec krvi. V zadnji vrstici imamo zopet oznako škatle ter pozicije v škatli.

Slika 3.1 Primerjava med nalepkami za različne vzorce. Zgoraj levo je nalepka nevrološke klinike, spodaj levo je nalepka izvornega vzorca, zgoraj desno je nalepka vzorca DNA, spodaj desno je nalepka vzorca krvi.

3.1.2 Kodni del nalepke

Kodni del nalepke sestavlja optična predstavitev unikatne označbe vzorca s kodo QR. Kode se med različnimi vzorci razlikujejo v nizu, ki je kodiran. Še eno prednost uporabe kod QR predstavlja dejstvo, da so lahko berljive z večino naprav, ki imajo vgrajen fotoaparat ali kamero in v tem primeru namenskega čitalca kod niti ne potrebujemo.

Specifikacija kode QR

Zaradi velikosti epruvet, v katerih so vzorci hranjeni, smo omejeni glede velikosti kode QR. Tako je bila največja koda, ki smo jo lahko natisnili na nalepko, v velikosti 21x21 slikovnih pik ter v celotni velikosti kode 7,5 mm. Predlagane so bile tudi kode v velikosti 5 mm in 10 mm, vendar se je pri prvi pokazalo, da so težko berljive brez namenskega čitalca kod. Koda velikosti 10 mm ni bila berljiva zaradi same specifikacije kode (koda nujno potrebuje belo obrobo široko vsaj 4 slikovne pike, za katere zaradi ukrivljenosti epruvete ni prostora).

Z velikostjo kode smo omejeni na 10 do 25 alfanumeričnih znakov, količina teh pa je odvisna od redundance kode same. Možne so 4 stopnje redundance 7 %, 14 %, 25 % in 30 %. Število znakov, ki jih lahko v kodo zakodiramo, pada z višjo stopnjo redundance. Tako lahko na najvišji stopnji redundance zakodiramo samo 10 alfanumeričnih znakov.

Zaradi lažjega branja ter identifikacije v primeru delne okvare kode smo se odločili za 16-mestno kodo s 25 % redundanco. Testirali smo tudi 10-mestno kodo s 30 % redundanco, vendar smo bili s tem zelo omejeni z unikatnim nizom označbe vzorca, ki pa ne odtehta 5 % korekcije napak.

Uporabili smo 16-mestno kodo QR, ki ima sledečo strukturo:

Prva tri mesta označujejo identifikator projekta, ki so še dodatno razdeljena na prvi znak, ki označuje laboratorij, ter dva znaka, ki označujeta projekt znotraj laboratorija. Naslednja tri mesta so namenjena razširitvam katerega od delov obstoječe strukture ali čisto nov podatek. Sledijo tri mesta, ki nam povedo vrsto vzorca.

- **000**: izvorni vzorec,
- **001**: vzorec DNA,
- **002**: prva rezerva vzorec krvi,
- **003**: druga rezerva vzorca krvi,
- **004**: vzorec DNA iz prve rezerve krvi (002),
- **005**: vzorec DNA iz druge rezerve krvi (003).

Zadnjih sedem mest označuje zaporedno številko vzorca. Zaporedna številka vzorca se avtomatsko inkrementira glede na vnos novega vzorca v podatkovno bazo.

Vsak izvorni vzorec je unikaten glede na projekt in zaporedno številko. Vsak njegov podvzorec pa je unikaten na projekt, vrsto vzorca in zaporedno številko. V primeru razširitve kodiranja vzorcev na druge projekte, bi zaradi strukture kodiranja morali obdržati identifikator projekta in zaporedno številko vzorca. Tako je sredinskih 6 znakov prostih za uporabo oznak vzorcev, ki bi jih projekt potreboval. S takim načinom lahko takoj razločimo kateremu projektu pripada in številko vzorca.

3.2 Konfiguracija strežnika

Ker je strežnik namenjen podpori tudi ostalim laboratorijskim procesom, njegove osnovne konfiguracije nismo spreminjali. Zaradi namestitve potrebne programske opreme smo naredili novega uporabnika in si poleg tega zagotovili sledljivost izvedenih ukazov ter s tem možnost prehoda nazaj v prejšnje stanje. Uporabnik je bil dodan v skupino administratorjev.

Poleg novega uporabnika smo naredili tudi sliko sistema celotnega virtualnega strežnika, kot še dodatno varnostno kopijo v primeru, če bi prišlo do kakšnih nepredvidenih zapletov pri nadaljnjih namestitvah. Slika sistema je bila po koncu kopirana na tračno enoto kot varnostna kopija.

3.3 Namestitev programske opreme

Programsko opremo smo namestili preko ukazne lupine Linux, s pomočjo programa *yum* (*angl. Yellowdog Updater, Modified*). Ukaz *yum* je bil zagnan preko ukaza *sudo*, ker privzeto naš uporabnik nima neomejenega dostopa do sistema. Poleg tega pa se vsaka uporaba ukaza *sudo* in njegovih rezultatov zabeleži v sistemske dnevnike.

Izpis 3.1 prikazuje 6 ukazov, ki smo jih izvedli pred začetkom namestitve programske opreme in tako preverili verzijo jedra operacijskega sistema Linux. S tem smo lahko poiskali zadnje verzije potrebovane programske opreme za našo verzijo operacijskega sistema. Strežnika se ni smelo posodobiti na zadnjo verzijo, saj nismo vedeli, kako bi s tem omejili ali onemogočili delovanje programske opreme. Celoten postopek posodobitve s preverjanejem delovanja, bi vzel precej dodatnega časa, s katerim nismo razpolagali.

Izpis 3.1 Seznam ukazov za namestitev programske opreme.

```
#izpise ime distribucije operacijskega sistema
$ cat /etc/*-release

#izpise verzijo operacijskega sistema
$ uname -a

#preko namestitvenega programa poisce program in izpise
#ime programskega paketa z moznimi verzijami
#yum search <ime_paketa>
$ yum search mysql
$ yum search mysql-server
$ yum search httpd
$ yum search php5
```

Najprej smo namestili spletni strežnik Apache. Privzeto se izvaja preko administracijskega uporabnika (*root*), vendar smo zaradi dodatne varnosti samega sistema in dostopa do tega naredili novega uporabnika, pod katerim se bo izvajal demon spletnega strežnika. Nov uporabnik ni imel neomejenega dostopa do samega datotečnega sistema. V nasprotnem primeru, bi imeli procesi spletnega strežnika možnost spreminjati katerikoli del datotečnega ali operacijskega sistema, saj bi imeli pravice administracijskega uporabnika. Uporabnika, pod katerim se izvaja spletni strežnik, smo ustvarili pred samo namestitvijo slednjega, saj se v primeru, ko uporabnik na sistemu že obstaja, izognemo dodatni konfiguraciji. Ker uporabnik že obstaja se namestitev programske opreme privzeto priredi prej ustvarjenemu uporabniku. Po končani namestitvi in pred zagonom spletnega strežnika, smo v konfiguraciji popravili privzeta vrata, preko katerih dostopamo do spletnega okolja. Spremembo smo naredili zaradi dodatne varnosti, saj v samem

začetku javni dostop do okolja ni bil predviden. Namestitev strežnika prikazuje izpis 3.2.

Izpis 3.2 Seznam ukazov pri namestitvi spletnega strežnika.

```
#ustvarimo skupino v katero bomo dodali uporabnika
$ sudo groupadd apache

#ustvarimo uporabnika in ga dodamo v skupino
$ sudo useradd -g apache apache

#zacetek namestitve spletnega streznika
#yum install <ime_paketa>
$ sudo yum install httpd

#z naslednjimi ukazi preverimo status demona
$ sudo service httpd status
$ sudo service httpd info

#popravimo konfiguracijsko datoteko
$ sudo vi /etc/httpd/conf/httpd.conf

#zazenemo demona spletnega streznika
$ sudo service httpd start
```

Namestitev programske opreme MySQL je potekala s privzetimi nastavitvami. Programska oprema je sestavljena iz dveh paketov, tj. iz strežnika in odjemalca. Odjemalec sicer ni potreben saj bo v veliki večini dostop do podatkovne baze poteka preko spletnega vmesnika. Vseeno pa je dobra rezerva za administracijo ter konfiguracijo samega relacijskega okolja in podatkovnih baz. Medtem ko strežnik poskrbi za to, da imamo preko mrežne povezave dostop do okolja, v katerem se nahajajo podatkovne baze. Namestitev programskega paketa MySQL prikazuje izpis 3.3.

Ukaz *mysql_secure_instalation* izvedemo po namestitvi in s tem programsko okolje še dodatno zaščitimo s preprostimi ukrepi kot so:

- dodeljevanje kompleksnega gesla administratorskem računu,
- odstranjevanje testne baze,
- preprečitev dostopa neavtoriziranim uporabnikom.

Vsi ti ukazi nam sicer precej poenostavijo začetno konfiguracijo okolja, vendar so v produkcijskem okolju bolj slabost kot prednost.

Izpis 3.3 Seznam ukazov pri namestitvi programskega paketa MySQL.

```
#namestitev odjemalca in streznika
#yum install <ime_paketa>
$ sudo yum install mysql mysql-server

#zagon demona
$ sudo service mysqld start

#zagon skripte za zascito streznika
$ sudo mysql_secure_installation
```

Na koncu namestimo še okolje za programski jezik PHP, v katerem bo poleg programskega jezika HTML napisana večina našega uporabniškega vmesnika. V jeziku PHP bo implementirano predvsem ozadje (angl. *backend*), ki se bo ukvarjalo s dostopom in manipulacijo podatkov v podatkovni bazi. Z jezikom HTML pa bo implementiran prikaz uporabniškega vmesnika. Namestiti smo morali tako programski jezik, kot tudi paket *php-mysql*, ki nam omogoča povezavo s podatkovno bazo MySQL. Poleg samega okolja smo namestili tudi programsko opremo *phpmyadmin*, ki je grafični vmesnik za manipulacijo okolja MySQL. Do vmesnika dostopamo preko spletnega brskalnika. Za osnovno administracijo zato ne potrebujemo dostopa do strežnika preko ukazne vrstice. Dostop do administracijskega vmesnika smo omejili na točno določene naslove IP, saj imajo do njega dostop samo pooblaščeni uporabniki. Poleg tega je omogočen lokalni dostop iz samega strežnika.

Po končani namestitvi je bilo potrebno preveriti, da se vsi demoni in servisi zaženejo ob zagonu samega sistema, da v primeru ponovnega zagona ne potrebujemo dodatnih administracijskih posegov. Namestitev prikazuje izpis 3.4.

Izpis 3.4 Seznam ukazov pri namestitvi programskega paketa PHP.

```
#namestitev programskega jezika in dosatnega paketa
#yum install <ime_paketa>
$ sudo yum install php php-mysql

#namestitev vmesnika za administracijo okolja
$ sudo yum install phpmyadmin
```

3.4 Struktura spletnega vmesnika

3.4.1 Struktura vmesnika na datotečnem sistemu

Datoteke, ki sestavljajo spletni vmesnik, se nahajajo v privzetem direktoriju spletnega strežnika, tj. */var/www/html/*. Vsak del vmesnika sestavljata dve datoteki.

Prva datoteka je sestavljena iz večinoma kode HTML in predstavlja izgled spletnega vmesnika. Datoteka je razdeljena na dva dela. V prvem imamo kodo CSS, ki določajo izgled posameznih elementov strani. Drugi del nam predstavlja masko za vnos ali prikaz podatkov. Vnos podatkov je implementiran preko *FORM* elementa HTML. Končna akcija preko ukaza HTTP *POST*, posreduje podatke v prej vpisana vnosna polja podatkovni bazi.

Drugo datoteko sestavlja programska koda PHP, preko katere posredujemo ali prikličemo podatke iz podatkovne baze. Programska koda skrbi za povezavo s podatkovno bazo. Preko nje podatke v podatkovno bazo vpišemo ali iz podatkovne baze preberemo. V primeru, ko zahtevamo podatke iz podatkovne baze, programska koda sestavi tudi stran za prikaz teh podatkov preko kode HTML.

Vmesnik je razdeljen na dve verziji, produkcijsko in razvojno. Produkcijska se nahaja v korenskem direktoriju spletnega strežnika, medtem ko se razvojna nahaja v poddirektoriju *dev/*. Dve ločeni okolji sta namenjena testiranju novih funkcionalnosti na razvojnem okolju, ki jih po tem prenesemo v produkcijsko okolje. S takim pristopom pri nadaljnjem razvoju ne motimo delovnega procesa in se lahko produkcijsko programsko opremo uporablja brez prekinitev.

Ko programske opreme aktivno ne razvijamo, sta okolji poenoteni (identični). Razlika je v vsebini, saj sta povezani na različni podatkovni bazi, saj produkcijskih podatkov ne smemo imeti v razvojnem okolju zaradi same varnosti.

Spletni vmesnik na datotečnem sistemu sestavljajo datoteke, ki so podane v izpisu 3.5.

Izpis 3.5 Datoteke, ki na datotečnem sistemu sestavljajo spletni vmesnik.

```
##sistemske in podporne strani
#struktura strani
header_html.php
header.php
index_html.php
index.php
login_html.php
login.php
menu_html.php
menu.php
#konfiguracija strani
cred.ini
userConnect.php

##strani za manipulacijo podatkov
#manipulacija novih vzorcev
generateNewSample_html.php
generateNewSample.php
insertNewSample_html.php
insertNewSample.php
#manipulacija obstojecih vzorcev
splitSubSample_html.php
splitSubSample.php
insertSubSample_html.php
insertSubSample.php
updateSubSample_html.php
updateSubSample.php
#prikaz zavedenih vzorcev
selectSample_html.php
selectSample.php
```

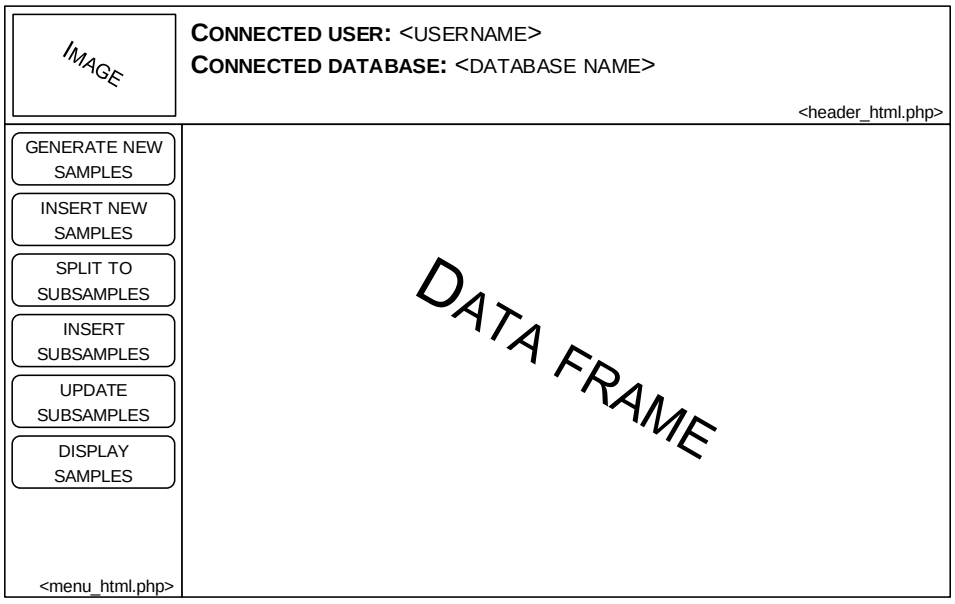
3.4.2 Struktura strani spletnega vmesnika

V naslednjem delu so predstavljene posamezne strani in opis njihovega delovanja. Vse strani se nahajajo tudi v razvojnem okolju. Ker je njihovo delovanje in struktura identična produkcijskim verzijam, le-te niso dodatno opisane.

Naslednje datoteke predstavljajo ogrodje samega spletnega grafičnega vmesnika. Potrebujemo jih za shranjevanje in prikazovanje informacij o uporabnikih ter podatkovnih bazah.

- **userCred.ini**: v datoteki so začasno vpisani uporabnikovi podatki, ki jih uporabljamo za uporanikov dostop do podatkovne baze. Namen datoteke je, da se izognemo odvečnim poizvedbam in s tem razbremenimo sistem z odvečnimi transakcijami. Datoteko sestavljajo uporabniško ime, geslo in trenutna podatkovna baza. Medtem, ko je uporabniško ime zapisano nekriptirano, pa je geslo zapisano kot kriptiran niz znakov.
- **userConnect.ini**: datoteka ustvari povezave na podatkovno bazo. Uporabljamo jo v vseh ostalih datotekah, ki povezavo na podatkovno bazo potrebujejo, ker je ta del programske kode skupen. S takim pristopom se izognemo napakam v delovanju, saj v primeru modifikacij popravljamo samo eno datoteko.
- **login_html.php in login.php**: datoteki vsebujeta programsko kodo, ki je potrebna za prijavo uporabnika. Programska koda preveri vpisano uporabniško ime in geslo s tabelo uporabnikov v podatkovni bazi. Če se podatki ujemajo, uporabnika obvestimo, da je prijava uspešna in ga preusmerimo na vstopno podatkovno stran. V nasprotnem primeru uporabnika obvestimo o napaki in ga preusmerimo nazaj na prijavno stran. Poleg same preusmeritve njegove podatke vpišemo v prej omenjeno datoteko userCred.ini. Kot smo omenili, je geslo zapisano v kriptirani obliki. Prav tako v tej obliki tudi preverjamo ujemanje gesel.
- **index_html.php in index.php**: datoteki predstavljata ogrodje spletnega vmesnika. Osnovno okno vmesnika je razdeljeno na 3 dele:
  - glavo spletnega vmesnika,
  - navigacijski meni,
  - podatkovni okvir.
- **header_html.php in header.php**: datoteki služita kot glava spletnega vmesnika z informacijami o prijavljenem uporabniku in povezani podatkovni bazi. Prikazani podatki so zapisani v datoteki *userCred.ini* in so zgolj informativne narave.

- **menu_html.php in menu.php**: v datotekah se nahaja programska koda navigacijskega menija. Navigacijski meni vsebuje šest akcij, preko katerih pridemo do različnih strani za manipulacijo zapisov podatkovne baze. Akcije so implementirane preko naslednjih gumbov (slika 3.2):
  - *Generate new samples*,
  - *insert new samples*,
  - *split to subsamples*,
  - *insert subsamples*,
  - *update subsamples*,
  - *display samples*.



Slika 3.2 Ogrodje spletnega vmesnika z vsemi elementi.

Naslednji sklop datotek predstavlja strani za manipulacijo in prikaz zapisov podatkovne baze. Struktura datotek je enaka kot prej, kjer končnica *_html.php* poredstavlja datoteko, ki vsebuje programsko kodo, ki se prikazuje v spletnem vmesniku, končnica *.html* pa datoteke s programsko kodo za interpretacijo podatkov iz prejšnje datoteke.

- **generateNewSample_html.php in generateNewSample.php**: s pomočjo te strani generiramo določeno število nizov, ki jih preko tiskalniške programske opreme zakodiramo v kodo QR in natisnemo na nalepke za evidentiranje vzorcev. Forma za vnos tako od uporabnika zahteva vnos števila vzorcev, ki jih hoče trenutno vnesti v bazo. Programski del preko prej podanega števila generira tolikšno količino novih nizov.

  Način generiranja nizov je naslednji:

  1. Preko metode *POST* preberemo kolikšno število nizov moramo generirati.
  2. Iz podatkovne baze preko poizvedbe SQL in uporabniških podatkov preberemo zadnji zavedeni vzorec in njegovo kodo.
  3. Glede na vrnjen niz generiramo število naslednjih zaporednih nizov.
  4. V datoteko, ki ima vrednosti ločene z vejico (v nadaljevanju datoteka CSV), zapišemo v prvo vrstico imena stolpcev, ter v vsako naslednjo niz, ki ga bomo zapisali na kodni del nalepke. Pogleg tega v vsako vrstico zapišemo še dele tega niza, ki se nahajata na tesktovnem delu nalepke.
  5. Na koncu pokličemo funkcijo, ki uporabniku ponudi avtomatsko shranjevanje kreirane datoteka na lokalni disk.

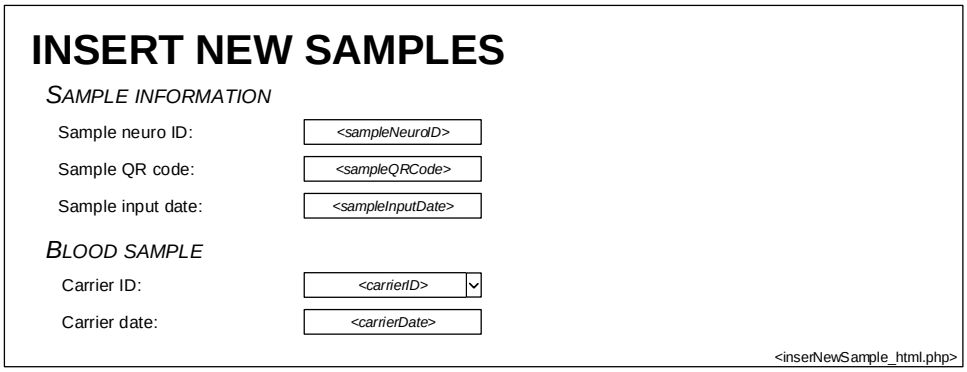
- **insertNewSample_html.php in insertNewSample.php**: stran nam omogoča vpisovanje vzorcev, katere smo generirali v podatkovno bazo. Stran sestavljata forma za vnos podatkov, ter programska koda, ki vpisane podatke interpretira in jih vstavi v podatkovno bazo. Zahtevani podatki za vnos so (slika 3.3):

  - *Sample neuro ID*,
  - *sample QR code*,
  - *sample Input date*.

  Poleg teh podatkov, pa potrebujemo tudi informativne podatke o dostavi vzroca, kot so:

  - *Carrier ID*,
  - *carrier date*.

  Pri vnosu podatkov v podatkovno bazo, preko poizvedbe SQL preverimo, če zapis s tem kliničnim identifikatorjem ali unikatno kodo že obstaja. Če zapis že obstaja uporabnika o

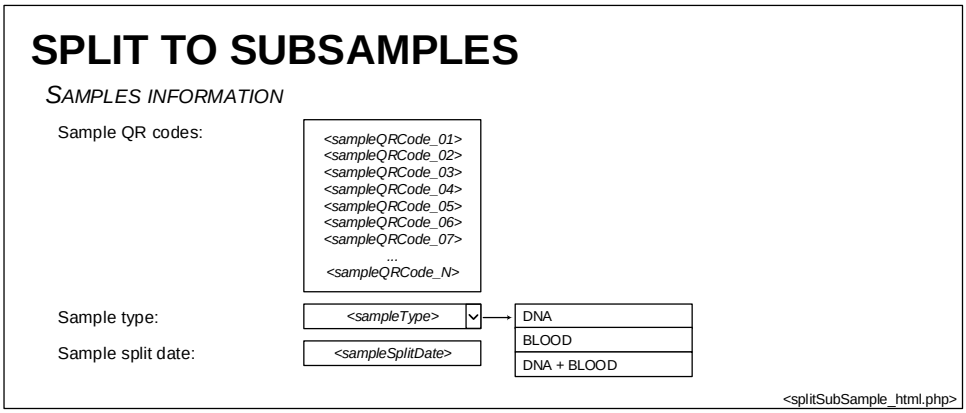


Slika 3.3 Stran za vnos novih vzorcev.

tem obvestimo in ne vstavimo ničesar, v nasprotnem primeru pa zapis vstavimo v bazo in uporabnika obvestimo o uspeli operaciji.

- **splitSubSample_html.php in splitSubSample.php**: v primeru, ko iz vzorca izoliramo DNA, se začetni vzorec razdeli na tri podvzorce (eden namenjen izolaciji DNA in dva vzorca krvi). Uporabnik mora vnesti naslednje podatke (slika 3.4):
  - *Sample QR codes*,
  - *sample type*.

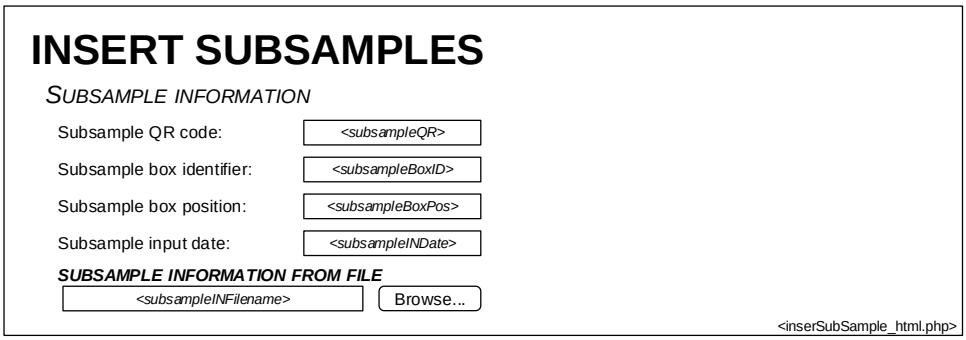
  Programska koda strani preko poizvedb SQL, pridobi podatke o izvornih vzorcih. Pregleda, da vsi vneseni vzorci v programski bazi obstajajo. Po tej operaciji, programska koda glede na vnesene vzorce in tip vzorca kreira datoteko CSV. Datoteka v prvi vrstici zopet vsebuje imena stolpcev, vsaka naslednja vrstica pa je sestavljena iz unikatne oznake vzorca (zakodiranega v kodo QR), del niza z informacijo o laboratoriju, zaporedno številko vzorca, tip vzorca, številko škatle (v kateri bo spravljen vzorec) in pozicijo vzorca v škatli (pozicije so označene od *A1* do *I9*). Številko škatle in pozicijo preračunamo glede na informacijo o zadnjem vzorcu v podatkovni bazi, katero pridobimo s pomočjo poizvedbe SQL. V primeru, da katera od novih generiranih pozicij v škatli presega prosta mesta, takrat inkrementiramo številko škatle ter začnemo pozicije beležiti od začetka. Na koncu uporabniku ponudimo avtomatsko shranjevanje datoteke.
- **insertSubSample_html.php in insertSubSample.php**: na tej strani vnesemo v podatkovno bazo podatke za vsak podvzorec posebej. Za vnos imamo dve možnosti. Prva je



Slika 3.4 Stran za razdelitev izvornih vzorcev na podvzorce.

vnos s pomočjo datoteke, ki smo jo dobili kot rezultat operacij na strani *splitSubSample_html.php*. Podatkov v bazo ne vpišemo že na prej omenjeni strani pri kreiranju datoteke v izogib napakam. Kot druga možnost pa za vsak vzorec vnesemo na formo za vpis naslednje podatke (slika 3.5):

- *Subsample QR code*,
- *subsample box identifier*,
- *subsample box position*,
- *subsample input date*.



Slika 3.5 Stran za vnos podatkov za podvzorce.

Pri potrditvi vnosa v podatkovno bazo, zopet preverimo, če zapisi že obstajajo, uporabnika

obvestimo o napaki in ne vstavimo ničesar. V primeru, da zapisi še ne obstajajo, uporabniku prikažemo kateri zapisi bodo vstavljeni, kjer mora uporabnik dodatno potrditi, da so podatki res pravilni (kot dodatni varnostni ukrep).

- **updateSubSample_html.php in updateSubSample.php**: za vzorec z izolirano DNA nas zanima kvaliteta in genotip, medtem, ko nas za vzorec krvi zanima količina in ali je vzorec še na voljo. Preko te strani lahko za vsak podvzorec vnesemo dodatne podatke (slika 3.6). V primeru vzorca DNA so to:
  - Atributi za kvaliteto DNA vzorca:

    Subsample A230 value,

    subsample A260 value,

    subsample A280 valuen,

    subsample A260/230 ratio,

    subsample A260/280 ratio,

    subsample concentration,

    subsample quality date.
  - Atributi za genotip DNA vzorca:

    Subsample filter01 temperature01, temperature02,

    subsample filter02 temperature01, temperature02,

    subsample genotype date.
  - Atributi vzorec krvi:

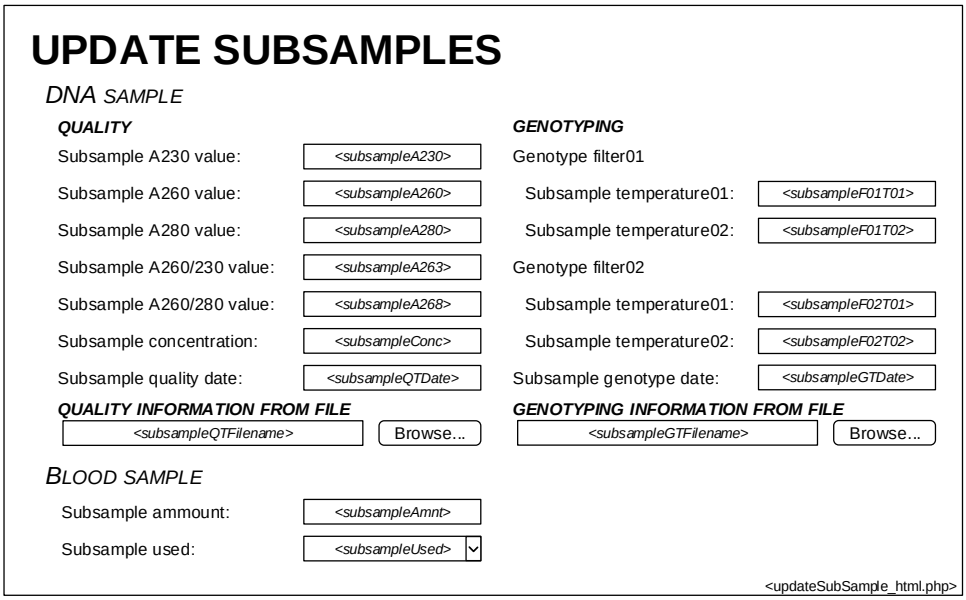
    Subsample ammount,

    subsample used.

- **selectSample_html.php in selectSample.php**: čeprav je akcija sestavljena iz dveh delov, se na vnosnem delu nahaja zgolj povezava na programski del, ki se zgodi ob kliku gumba. Programski del tako naredi poizvedbo v podatkovno bazo in nato izpiše vse zapise z vsemi stolpci v podatkovni bazi.

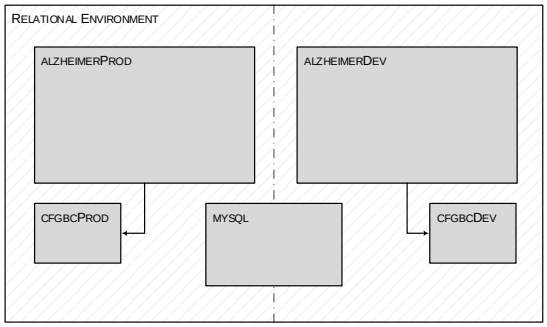
3.5 Struktura podatkovne baze

Podatkovna baza je realizirana v relacijskem okolju MySQL. Nameščeni imamo dve podatkovni bazi, tj. razvojno z imenom *alzheimerDev* in produkcijsko *alzheimerProd* (slika 3.7). Po struk-



Slika 3.6 Stran za vnos dodatnih podatkov za podvzorce.

turi sta podatkovni bazi identični, razlikujeta se po vsebini, saj razvojna ne vsebuje realnih podatkov. Razvojna podatkovna baza je tako namenjena testiranju implementacije novih funkcij in preverjanju pravilnosti delovanja le-teh.

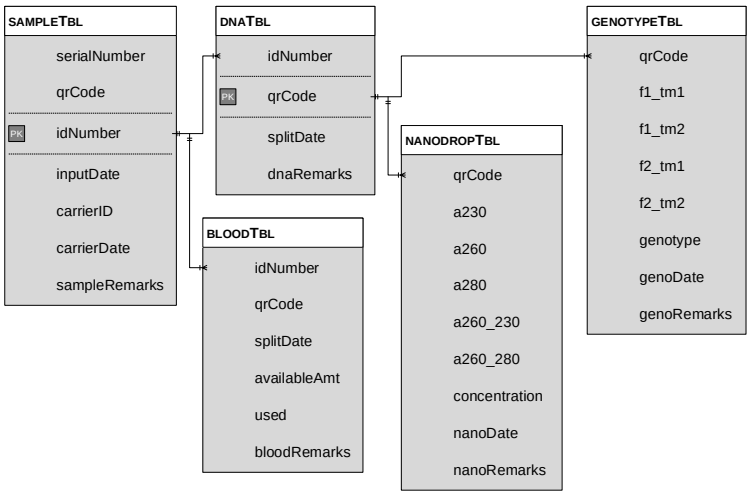


Slika 3.7 Struktura relacijskega okolja.

Razlikujeta se tudi v uporabnikih, ki so definirani nad bazama. Uporabniki so razdeljeni v več skupin, katere imajo definirane svoje pravice in omejitve dostopa do podatkovnih baz. Najširši nabor skupin je definiran nad razvojno bazo.

- *Readers*: uporabniki imajo pravico pregledovanja podatkov, lahko pa tudi vsakemu uporabniku dodatno omejimo dostop samo do določenih pogledov ali tabel.
- *Editors*: standardni uporabnik raziskovalca v laboratoriju. Uporabnik ima pravice branja in vnašanja podatkov v podatkovno bazo.
- *Developers*: uporabniki te skupine spadajo pod razvijalce programske opreme, kateri razvijajo nove funkcionalnosti ali odpravljajo napake. Poleg standardnega uporabnika imajo tudi pravico brisanja vsebine in ustvarjanja novih tabel ter podatkovnih baz.
- *Administrators*: uporabniki te skupine imajo vse pravice nad relacijskem okoljem (tudi brisanje sistemskih podatkov in tabel). Uporabnik *root*, je sicer obravnavan kot administrativni uporabnik, vendar je edini, ki ima neomejeno dostop (najvišji administrativni uporabnik).

Vse skupine, razen *Developers*, so definirane tudi na produkcijski podatkovni bazi. Ta ni potrebna, saj nad produkcijsko podatkovno bazo ne izvajamo razvoja in implementacije novih funkcionalnosti. Prav tako smo onemogočili anonimen dostop do relacijskega okolja (brez uporabniškega imena in gesla je dostop onemogočen). Dostop je bil implementiran na ta način zato, da preprečimo dostop do sistemskih tabel nepooblaščenim uporabnikom.



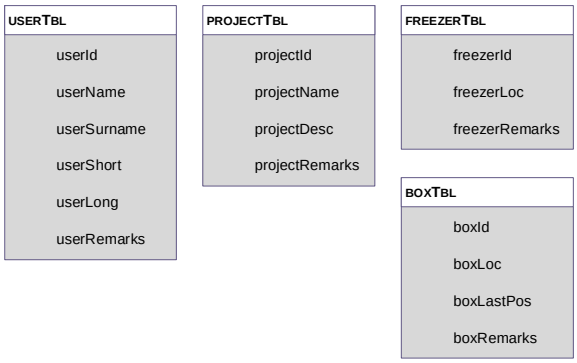
Slika 3.8 Struktura osnovnih tabel podatkovne baze.

Podatkovno bazo sestavlja več tabel. Osnova so tabele, v katerih so zavedeni vzorci in podvzorci (slika 3.8). Zaradi lažje preglednosti in upravljanja s podatki, so tabele smiselno razdeljene

glede na stopnjo vzorca v raziskovalnem postopku.

Poleg osnovnih tabel vzorcev imamo tudi podporne in sistemske tabele:

- Sistemske tabele (nahajajo se v sistemski podatkovni bazi *mysql*, na sliki 3.7):
  - **users**: predstavlja sistemsko tabelo uporabnikov. V njej so definirani vsi uporabniki in skupine uporabnikov, ter njihove dostopne pravice (do katerih podatkovnih baz imajo dostop, ter katere akcije lahko vršijo nad podatkovnimi bazami).
- Podporne tabele (nahajajo se v podatkovni bazi *cfgbcProd*, prikazane na sliki 3.9):
  - **userTbl**: v tabeli so zavedeni laboratorijski uporabniki, ki imajo dostop do podatkovnih baz. Pravilnost dostopnih podatkov se sicer preverja preko sistemske tabele *users*, trenutno pa se uporablja kot podporna tabela za prikazovanje in povezovanje uporabniških podatkov (polno ime, opomb itd.).
  - **projectTbl**: podporna tabela za razpoznavanje identifikatorjev projektov. Prav tako, pa so v tabeli zavedena imena, opisi in opombe za posamezen projekt.
  - **freezerTbl**: tabela hladilnikov in zamrzovalnikov. V tabeli so zavedene lokacije in identifikatorji posameznih hladilnikov, saj so škatle v katerih shranjujemo vzorce shranjene v njih.
  - **boxTbl**: v tabeli so shranjeni podatki o škatlah, v katerih hranimo vzorce. Za vsako škatlo imamo informacije o identifikatorju, lokaciji, zadnji prosti poziciji in opombe. Lokacijo škatle hranimo kot identifikator hladilnika.



Slika 3.9 Podporne tabele.

4 Postopek evidentiranja vzorca

4.1 Evidentiraje novih vzorcev

Izvorne vzorce prevzamemo v laboratoriju na Nevrološki kliniki (NK). Slednji so že opremljeni z nalepko na kateri je štirimestna številka s kodo QR. Potrebno je izpolniti evidenco NK, da smo vzorce prevzeli in evidenco CFGBC, da so vzorce oddali. Evidenčni podatki vsebujejo datum, številke vzorcev, ter podpisa obeh udeleženih oseb. Ko vzorec ali vzorce pooblaščena oseba prinese v laboratorij CFGBC, se na delovni postaji namenjeni kreiranju evidenčnih nalepk, generira nalepke za toliko novih vzorcev kolikor smo jih prinesli. Proces kreiranja novih kod QR in tiskanja le-teh je delno avtomatiziran. Spletni vmesnik nam izdela datoteko z vrednostmi, katero potem uvozimo v program za tiskanje. Ta nam na podlagi prej ustvarjene datoteke in predloge strukture nalepke, le-te natisne. Natisnjene nalepke nato nalepimo na prinešene vzorce. Ker par zaporedne številke in kode QR še ni zaveden v bazi, vrstni red lepljenja nalepk ne igra vloge.

4.2 Vnos vzorca v bazo

Ko imamo na vseh vzorcih nalepko z zaporedno številko in nalepko s kodo QR, nadaljujemo z vnosom vzorcev v podatkovno bazo. Na delovni postaji, kjer smo prej ustvarili nalepke, je tudi namenski čitalec kod. Z njegovo pomočjo preko spletnega vmesnika vzorce vnesemo v podatkovno bazo. Na strani za vnos novih vzorcev se najprej vnese zaporedno kodo, tako da s čitalcem preberemo kodo QR zaporedne številke. Nato v naslednje polje za vnos spet preko čitalca preberemo kodo QR vzorca, katero smo generirali v prejšnjem koraku. Ročno vnesemo še datum (ki je sicer privzeto programsko nastavljen na trenutni datum), vnosa vzorca v podatkovno bazo. Naslednji sklop podatkov na vnosni strani zahteva tudi osnovne podatke o vzorcu. To sta datum prevzema vzorca in identifikator (ime) osebe, ki je vzorec prevzela. Ker je vzorec brez vnosa v bazo lahko hranjen dalj časa v hladilnikih, datum prevzema vzorca ni nujno enak datumu vnosa v podatkovno bazo. Prav tako podatek o uporabniku ni nujno enak uporabniku, ki vzorce vnaša v podatkovno bazo.

Na koncu se podatke pregleda, ter vnos potrdi. Ko so vsi trenutni vzorci zavedeni v bazo, se jih shrani v hladilnik s temperaturo –80 °C. Vzorce hranimo dokler se jih ne nabere dovolj, da je smiselno narediti izolacijo DNA.

4.3 Razdeljevanje vzorca pri izolaciji DNA

Ko se nabere dovolj vzorcev, vzorce vzamemo iz hladilnika ter jih pripravimo za izolacijo DNA. Izolacija se največkrat dela na osmih vzorcih naenkrat zaradi obvladljivosti samega postopka ter omejitev same laboratorijske opreme. Vzorce iz –80 °C segrejemo na 22 °C. Ker postopek traja nekaj časa, si lahko med tem pripravimo evidenčne nalepke za nove vzorce, ki jih pridobimo pri izolaciji DNA. Iz 10 ml epruvete vzorec razdelimo na 4 ml vzorec, ki ga uporabimo za izolacijo DNA, ter na dva 2 ml vzorca krvi, ki jih shranimo na –80 °C za rezervo, če izolacija ne bi bila uspešna, ali za namene kakšne druge raziskave.

Preko spletnega vmesnika v polje za vnos s pomočjo optičnega čitalca vnesemo kode QR vzorcev, ki smo jih izbrali za izolacijo DNA. Poleg tega moramo izbrati tudi, katere tipe vzorcev bomo generirali (DNA, kri ali oboje). Možnosti se razlikujejo samo v količini nalepk, ki jih naenkrat natisnemo. Vmesnik nam ustvari datoteko z vrednostmi, na podlagi katerih nam program za tiskanje s pomočjo šablone (tokrat namenjena podvzorcem), natisne nalepke s podatki in kodami QR posameznih vzorcev. Nalepko za identifikacijo vzorca DNA, shranimo do končanega postopka izolacije saj je nalepka namenjena označevanju končne epruvete (centrifugirke), v kateri

shranjujemo produkt celotnega postopka (raztopljena DNA). Samo končno epruveto označujemo zato, ker se v postopku epruvete velikokrat zamenjajo.

Vzorec DNA vnesemo v podatkovno bazo po končanem postopku izolacije. Vzorce vstavimo v pripadajočo škatlo in shranimo v hladilnik pri 4 °C.

Rezervna vzorca krvi označimo takoj, saj jih je potrebno čim prej shraniti v hladilniku na –80 °C, ker večkratno odmrzovanje in zamrzovanje škodi kvaliteti krvi. Po označevanju vzorca (ali vzorec, če je izvorne krvi premalo) zavedemo v podatkovno bazo, tako da s čitalcem preberemo kodo QR in poleg tega zavedemo še količino vzorca.

4.4 Vnos rezultatov meritev koncentracije

Ker nas zanima uspešnost in kakovost izolacije vzorca, to preverimo z meritvami na obstoječi laboratorijski opremi. Za lažji in hitrejši vnos vzorcev, optični čitalec premaknemo na delovno postajo, na kateri izvajamo meritve. Označevanje vzorcev z njihovimi identifikatorji nam olajša tudi vnos podatkov meritve kvalitete v podatkovno bazo.

V programsko opremo najprej preberemo kodo QR z namenom označitve vzorca. Nato iz centrifugirke vzamemo manjšo količino (par μl) raztopljene DNA ter na opremi izmerimo njeno kvaliteto. Po končani meritvi se podatki avtomatsko zavedejo v programsko opremo, tako da lahko nadaljujemo z merjenjem novega vzorca.

Po končanih meritvah vzorcev lahko iz programske opreme izvozimo tekstovno datoteko s podatki. Slednjo lahko preko spletnega vmesnika vnesemo v podatkovno bazo, če smo v prejšnjem koraku za označitev vzorca vnesli unikatni identifikator vsakega vzorca, ki je kodiran v kodi QR. Vnos podatkov preko datoteke vsem vzorcem, ki smo jih izmerili, vnese rezultate meritev kvalitete. Preden se podatki dokončno zavedejo v podatkovni bazi nam spletni vmesnik vrne podatke, ki bodo vneseni, da jih uporabnik lahko preveri in nato potrdi vnos. Možen je tudi ročni vnos, vendar se v tem primeru vnaša rezultate za vsak vzorec posebej.

Po vnosu rezultatov, se preveri ali bi bilo potrebno izolacijo katerega vzorca ponoviti (na podlagi kvalitete).

4.5 Vnos rezultatov genotipizacije

Pri procesu določanja genotipa (genotipizacija), uporabljamo del vzorcev DNA, ki smo jih v prejšnjem koraku izolirali. Manjše količine vzorcev (od 5 do 10 μl) prenesemo na ploščico, ki jo potem vstavimo v napravo za genotipizacijo (*LightCycler 480*). Na ploščico lahko nanesemo

do 384 različnih vzorcev. Napravo upravljamo preko programske opreme, ki je nameščena na pripadajoči delovni postaji.

Sam postopek eksperimenta določimo v samem programu. Programski opremi moramo tudi določiti, kje se kateri vzorec na ploščici nahaja. Za ta postopek imamo dve možnosti. Lahko preko grafičnega vmesnika programa ročno vnesemo identifikator vzorca, kjer se ta nahaja. Pozicije na ploščici so označene z matriko od *A1* do *P24*. Za vsako pozicijo lahko vnesemo identifikator vzorca, opombe ter določimo tip. Tipi med katerimi lahko izbiramo so vzorec, pozitivna kontrola ali negativna kontrola. Opcijsko pa lahko izberemo tudi barvo, zgolj kot vizualni identifikator. Kot drugo možnost, lahko s pomočjo predloge ustvarimo datoteko CSV s prej omenjenimi podatki, ki jo nato uvozimo v programsko opremo, ki nam na podlagi le-teh pripravi matriko vzorcev.

Po končanem eksperimentu lahko podatke, ki so bili izmerjeni za posamezen vzorec izvozimo v datoteko CSV. Preko te lahko rezultate vnesemo v podatkovno bazo. Vnos lahko izvedemo tudi ročno, vendar pri tem lahko vnesemo samo en vzorec naenkrat.

4.6 Določanje genotipov glede na rezultate genotipizacije

Pri genotipizaciji pridobi vsak vzorec DNA še do štiri dodatne podatke. Za vsakega od dveh različnih filtrov lahko dobimo dve različni temperaturi. S filtri merimo dve različni mesti v zaporedju, medtem ko temperaturi dobimo glede na tališča sond. Rezultate potem uporabimo pri interpretaciji in določanju genotipa.

Glede na višine temperatur tališč lahko dobimo eno ali dve različni temperaturi. V tabeli 4.1 so podane vse možne kombinacije temperatur pri različnih filtrih za iskane genotipe.

ApoE tip	E3/E3	E2/E2	E2/E3	E2/E4	E3/E4	E4/E4
T_m pri 530	55 °C	55 °C	55 °C	55°C/64 °C	55 °C/64 °C	64 °C
T_m pri 640	63 °C	53 °C	53 °C/63 °C	53 °C/63 °C	63 °C	63 °C

Tabela 4.1 Kombinacija temperatur za določanje genotipa.

Določitev genotipa glede na izmerjene temperature se izvede avtomatično pri vpisu rezultatov genotipizacije v podatkovno bazo. Takrat se preko enostavne programske kode PHP in pogojnih stavkov za vsak vzorec glede na vhodne podatke določi genotip. Tega se tudi zapiše v podatkovno bazo.

4.7 Kreiranje končnih rezultatov in izvidov

Na koncu raziskave se rezultati le-te predstavijo v obliki poročila oziroma izvida. Preden se lahko izvid izda in se jamči glede predstavljenih rezultatov, mora meritve ter njihove rezultate skozi celoten potek raziskave pregledati poleg samega raziskovalca tudi vnaprej določena nadzorna oseba. V primeru, da rezultati kakšnega vzorca preveč odstopajo, se lahko odloči, da se celoten ali del meritev za vzorec ponovi. Ko raziskovalec in nadzorna oseba meritve in rezultate potrdita, se lahko izdela izvid za vsak vzorec posebej.

Izvid se predstavi nevrološki kliniki z interpretacijo rezultatov. Interpretacija nam pove kakšen genotip je bil določen za posamezen vzorec ter kolikšno (kolikokrat povečano) je predvidevano tveganje glede na določeni genotip za raziskovano bolezen.

5 Sklepne ugotovitve

V času trajanja diplomskega dela, smo v celoti razvili in implementirali podporni informacijski sistem za evidenitranje in shranjevanje podatkov bioloških vzorcev. Poleg samega informacijskega sistema smo razvili tudi sistema standardnega označevanja vzorcev, ki temelji na unikatnem označevanju kodiranem v optičnih kodah.

Preko celotnega procesa razvoja informacijskega sistema smo prišli od razdrobljenega in nesistematičnega do centraliziranega, strukturiranega in varnega shranjevanja podatkov o vzorcih. Najprej smo morali vpeljati standard evidentiranja vzorcev in določiti podatke, ki jih je potrebno za vsak vzorec zavesti. Standard smo morali vpeljati, ker relacijske podatkovne baze delujejo na strukturiranih podatkih (podatki ustrezajo vnaprej določeni strukturi). Podatki, kateri so nujni za vsak vzorec, pa nam pomagajo hitro razlikovanje med več vrstami vzorcev. To je še posebej pomembno, če hranimo vzorce za več projektov. Uvedli smo tudi označevanje vzorcev z nalepkami, na katerih je vsak vzorec označen s svojim unikatnem kodiranjem QR. Optično kodiranje je pomembno za hitro vpisovanje kompleksnih nizov, ki predstavljajo unikaten identifikator vzorca. Prav tako pomaga tudi pri pridobivanju podatkov za posamezen vzorec iz podatkovne baze. Struktura nalepke je enotna za vse vzorce, ki jih imamo evidentirane v informacijskem

sistemu. Implementirati smo morali sistem shranjevanja podatkov preko uporabe relacijskega okolje *MySQL*. Razvili smo svoj grafični uporabniški vmesnik, za katerega smo uporabili spletno tehnologijo *HTML* v povezavi s programskim jezikom *PHP* in spletnim strežnikom *Apache*. Dostop do relacijskega okolja je realiziran preko uporabniškega vmesnika in ga uporabljamo za interakcijo s podatkovno bazo. Interakcija predstavlja vpisovanje, spreminjanje, prikazovanje, uvoz in izvoz podatkov. Uvoz in izvoz sta implementirana preko standardnih datotek *CSV*, saj je to format, katerega podpira večina naprav s katerimi opravljamo meritve in analizo vzorcev. Razvoj je vključeval tudi konfiguracijo obstoječe in novo nameščene programske opreme. Konfiguracija nam je zagotovila medsebojno kompatibilnost, povezljivost in varnost. Varnostne kopije v primeru fizičnih ali sistemskih napak smo realizirali na nivoju datotečnega sistema, s kopijo trdega diska virtualnega strežnika in periodičnih izvozov podatkovne baze. Sistem smo zaščitili z omejitvijo dostopa do strežnika in podatkovne baze ter avtorizacijo uporabnikov preko uporabniškega vmesnika. Celoten sistem je bil razvit skalabilno in modularno. Programsko opremo je enostavno prilagoditi v primeru povečanega obsega podatkov ali nadgradenj. Podatkovne baze so lahko prenosljive, zaradi virtualizacije pa to drži tudi za strežnik navkaterem temelji naš informacijski sistem. Sistem je bil načrtovan kot ogrodje v katerem vsak posamezen projekt predstavlja svoj modul in bi bilo vključevanje novih modulov manj zamudno.

Za prihodnost informacijskega sistema, so možne dodelave in nadgradnje na kateregakoli od njegovih delov. Lahko izpostavimo nekaj pomembnejših.

- **Varnost**:
  - avtorizacija: smiselno bi bilo razviti boljši sistem avtorizacije uporabnikov. To je še posebej pomembno v primeru, da bi se informacijski sistem širil in služil tudi kot povezava z drugimi raziskovalnimi ustanovami,
  - kriptiranje omrežnega prometa: v primeru, da bi komunikacija s podatkovno bazo potekala izven lokalnega omrežja, bi bilo dobro omrežni promet kriptirati zaradi možnega zajemanja omrežne komunikacije in posledičnih napadov na sistem.
- **Modularnost**:
  - vpeljava ogrodja za module: lahko bi razvili ogrodje za module, ki jih naš sistem vsebuje. Tako bi lahko določili kritične komponente, ki jih mora modul vsebovati za lažjo in enostavnejšo implementacijo le-tega v sistem.

- **Namestitev**:
  - avtomatizacija namestitve: vso programsko opremo, ki jo potrebuje naš sistem za delovanje vi lahko dodali v večji paket, ki bi nam omogočal enostavnejšo namestitev, brez večjih ali kompleksnejših ročnih posegov,
  - združevanje konfiguracije: konfiguracijske datoteke programske opreme bi lahko združili in se tako izognili razdrobljenosti in nepreglednosti.
- **Grafični vmesnik**:
  - izgled: uporabniški vmesnik je bil razvit s poudarkom na funkcionalnosti. Lahko bi ga nadgradili z dodatnimi funkcijami, ga naredili bolj preglednega in prijaznejšega za uporabo.

Sistem je trenutno v produkcijski uporabi na Centru za funkcijsko genomiko in bio-čipe. Pripomogel je k lažjemu označevanju in evidentiranju izvornih vzorcev, saj postopki vključujejo enostavne, delno avtomatizirane korake z malo ročnega vnašanja podatkov. Shranjevanje vzorcev v hladilnikih je bolj sistematično, zaradi tega lahko predhodno preko podatkovne baze izvemo lokacijo vzorca in nam ga ni treba pretirano iskati. Priprava eksperimentov, meritev, končnih poročil in vpisovanje rezultatov je manj zamudno zaradi izvoza in uvoza tekstovnih predlog iz podatkovne baze.

LITERATURA

[1] S. Klocke, A. Schierholz, *Thermal transfer printer*, US Patent App. 29/400,030 (Jul. 30 2013).

[2] J. Long, R. B. Moreland, *Direct thermal printer*, US Patent 6,784,906 (Aug. 31 2004).

[3] T. J. Soon, *QR code*, Synthesis Journal 2008 (2008) 59–78.

[4] W. M. Watkins, H. W. Lin, K. McClelland, R. A. Ullrich, S. Khanjenoori, K. Dalton, A. T. Lai, M. Kuca, S. Pacheco, J. Shaffer-Gant, *COTS software selection process*, Tech. rep., Sandia National Laboratories (2006).

[5] V. Tchantchaleishvili, J. D. Schmitto, *Preparing a scientific manuscript in Linux: Today's possibilities and limitations*, BMC research notes 4 (1) (2011) 434.

[6] R. Bowen, K. Coar, *Apache Cookbook: Solutions and Examples for Apache Administration*, "O'Reilly Media, Inc.", 2007.

[7] A. Trachtenberg, D. Sklar, *PHP Cookbook: Solutions and examples for PHP Programmers*, "O'Reilly Media, Inc.", 2006.

[8] M. A. Mohamed, O. G. Altrafi, M. O. Ismail, *Relational vs. NoSQL databases: A survey*, International Journal of Computer and Information Technology 3 (03) (2014) 598–601.

[9] P. DuBois, *MySQL Cookbook: Solutions for Database Developers and Administrators*, "O'Reilly Media, Inc.", 2014.

[10] F. F. Ozair, N. Jamshed, A. Sharma, P. Aggarwal, *Ethical issues in electronic health records: A general overview*, Perspectives in clinical research 6 (2) (2015) 73.