SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

RAZVOJ SUSTAVA ZA DUGOTRAJNO PRAĆENJE RADA TELESKOPA MAGIC/CTA

Ante Rota

Mentor: Nikola Godinović

Split, prosinac 2016.

S V E U Č I L I Š T E U S P L I T U

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Diplomski studij: Elektronika i računalno inženjerstvo

Smjer/Usmjerenje: Elektronika

Oznaka programa: 221

Akademska godina: 2016./2017.

Ime i prezime: Ante Rota

Broj indeksa: 679-2014

ZADATAK DIPLOMSKOG RADA

Naslov: Razvoj sustava za dugotrajno praćenje rada teleskopa MAGIC/CTA

Zadatak: Konkretan doprinos razvoju programske podrške sustava dugotrajne provjere kvalitete podataka MAGIC/CTA (Cherenkov Telecope Array) teleskopa. Sustav provjere kvalitete podataka je vrlo važan dio svakodnevnog rada MAGIC teleskopa i buduće mreže gama-teleskopa CTA. Povratna informacija o mogućim problemima je ključna u otklanjanju poteškoća u radu ovih teleskopa koji mjere najenergetskiji dio elektromagnetskog spektra - gama područje.

Prijava rada: 05.10.2016.

Rok za predaju rada: 20.02.2017.

Rad predan: 19.12.2016.

Predsjednik

Odbora za diplomski rad:

Mentor:

Prof. dr. sc. Julije Ožegović

Izv. prof. dr. sc Nikola Godinović

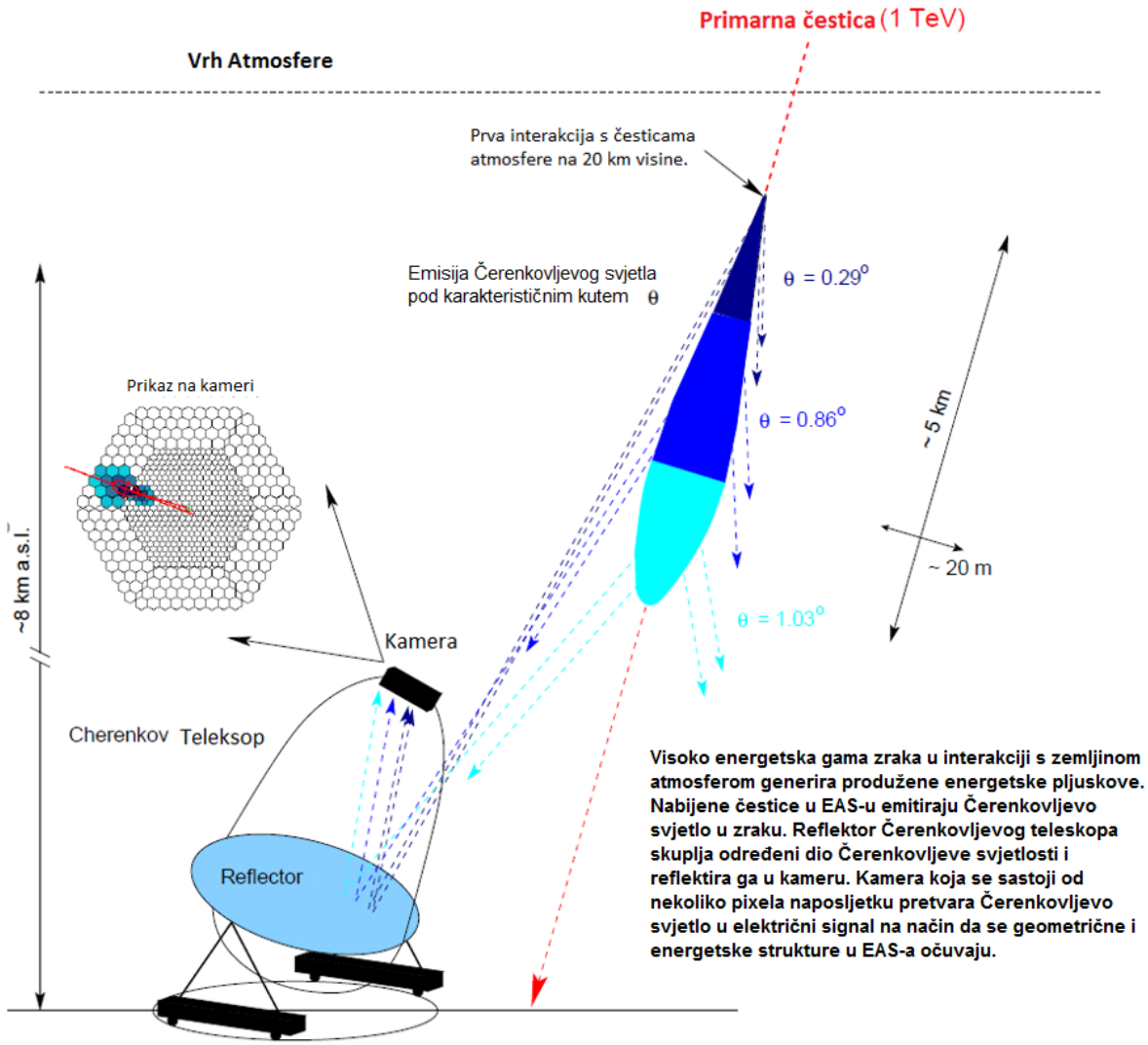
Sadržaj

1. Uvod

Jedna od najuspješnijih grana astročestične fizike danas je gama astronomija. Gama astronomija istražuje kozmičke gama zrake, odnosno elektromagnetsko zračenje najviših energija koje dolazi iz Svemira. Niže energijsko područje kozmičkih gama zraka (do par desetaka GeV) dostupno je, neposredno, samo detektorima na satelitima. Više energijsko područje kozmičkih gama zraka (od par desetaka GeV do par desetaka TeV) proučava se posredno detektorima na Zemlji. Visokoenergijske gama zrake (kao i viskoenergijske nabijene čestice, takozvane kozmičke zrake) izazivaju u atmosferi pljuskove sekundarnih čestica, elektrona i pozitrona. Nabijene sekundarne čestice u pljusku, čija je brzina veća od brzine svjetlosti u zraku, emitiraju posebnu vrstu elektromagnetskog zračenja - Čerenkovljevo zračenje. Čerenkovljevo zračenje iz pljuska sekundarnih čestica u atmosferi moguće je opaziti posebnom vrstom teleskopa.

1.1. Imaging Atmospheric Cherenkov Technique (IACT) metoda detekcije gama zraka

Pljusak sekundarnih čestica, elektrona i pozitrona, izazvanih upadnom gama zrakom, pri prolasku kroz zemljinu atmosferu će zračiti Čerenkovljevo svjetlo. Plavičasti, par nanosekundni, bljesak Čerenkovljevog svjetla putuje do teleskopa, te se reflektira pomoću zrcala teleskopa i fokusira se u kameri. Kamera, koja se u slučaju MAGIC teleskopa, sastoji od oko tisuću foto-multiplikatora pretvara fotone svjetla u električni signal koji se naknadno obrađuje. Proces je prikazan na idućoj slici (slika 1).



Slika 1: Ilustracija IACT detekcijske metode [1]

1.2. Cherenkov Telescope Array (CTA)

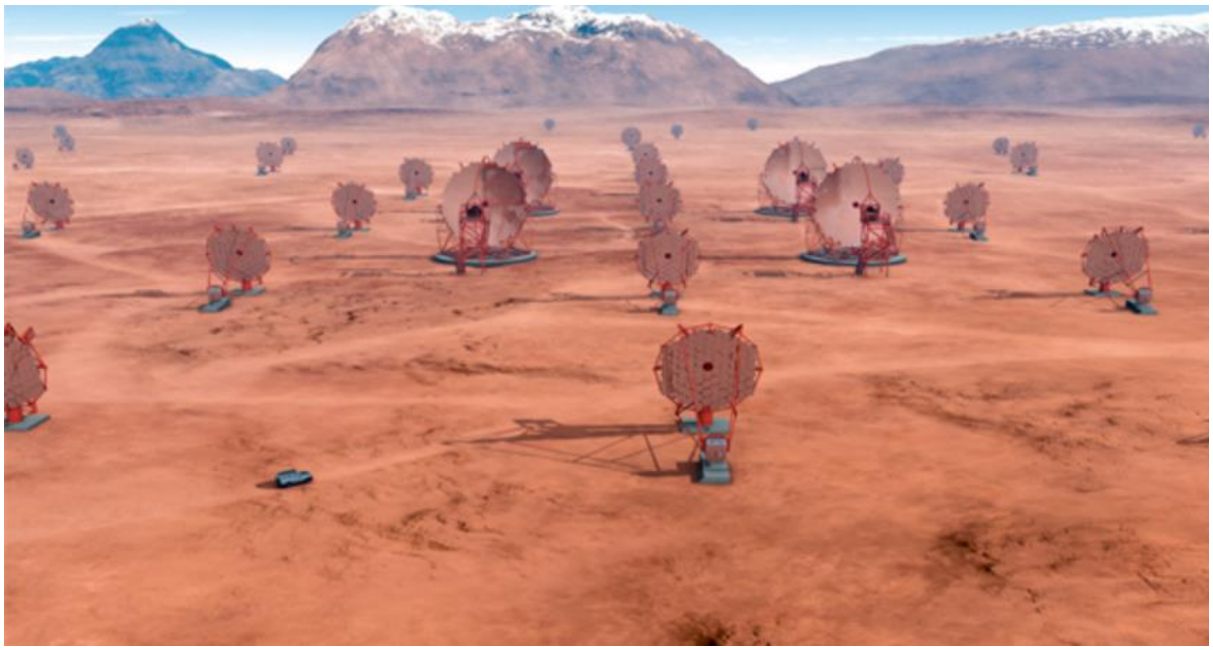
CTA [2] je projekt izgradnje iduće generacije zemaljskih gama teleskopa. Koristit će se kao otvoreni opservatorij za široku zajednicu astrofizičara i davati detaljan uvid u ne termalne procese visokih energija u svemiru.

Ciljevi CTA se mogu podijeliti u 3 skupine:

- razumijevanje podrijetla kozmičkih zraka i njihova uloga u svemiru,
- razumijevanje mehanizama nastajanja gama zračenja u svemiru,
- traženje konačne prirode i ponašanja elementarnih čestica i fizike van Standardnog modela elementarnih čestica.

Sadašnja generacija Čerenkovljevih teleskopa (H.E.S.S., MAGIC i VERITAS) u zadnjih nekoliko godina otvorila je uvid u astronomiju gama zraka u energetskim razredima poviše nekoliko desetaka GeV. CTA će detaljno istražiti naš Svemir u visoko energetskim razredima (E > 10 GeV) gama zračenja, te istražiti kozmičke netermalne procese, u bliskoj suradnji s opservatorijima koji promatraju Svemir na drugim valnim duljinama elektromagnetskog spektra, kao i koristeći druge nositelje informacija, kao što su kozmičke zrake i neutrini.

CTA (slika 2) će imati velik potencijal za istraživanja u ključnim područjima astronomije, astrofizike i osnovnih istraživanja u fizici. Ovo uključuje istraživanje podrijetla kozmičkih zraka i njihovog doprinosa u sadržaju svemira, istraživanje prirode i raznolikosti svemirskih akceleratora čestica, istraživanje crnih rupa, ispitivanje krajnje prirode materije i fizike nakon Standardnog modela, proučavanje tamne materije i njenog efekta na kvantnu gravitaciju.



Slika 2: Prikaz buduće instalacije CTA.

Dizajn ovih teleskopa predviđa poboljšanje od 5 do 10 puta u preciznosti u odnosu na trenutačne gama teleskope, te povećanje dostupnog spektra energije od oko 10 GeV do iznad 100 TeV.

1.3. Znanost iza CTA projekta

Radijacija visokoenergetskih gama zraka znatno se razlikuje od one koja je zabilježena na niskim energijama. Gama zrake s energijama od GeV do TeV ne mogu se uvjerljivo generirati termalnom emisijom vrućih objekata u svemiru. Energija termalne radijacije je ovisna o temperaturi tijela koji je emitira. Osim velikog praska ne postoji ništa dovoljno zagrijano u poznatom svemiru da emitira takve gama zrake.

Umjesto toga, nalazimo visokoenergetske gama zrake u hladnome svemiru, gdje su drugi mehanizmi odgovorni za koncentraciju velike količine energije u jednom kvantu zračenja.

Uobičajeno je objašnjenje da se gama zrake generiraju u interakciji s česticama ubrzanih do relativističkih brzina (brzine koje su blizu brzine svjetlosti). To se događa, primjerice, kada se čestice ubrzaju u valnim udarima koji nastaju pri eksploziji zvijezda, ili pri interakciji s magnetskim poljima. Tok i energetski spektar gama zraka reflektira tok i energiju visoko energetskih čestica. To znači da se gama zrake mogu koristiti za proučavanje mehanizama nastanka kozmičkih zraka i čestične strukture u dalekim područjima naše galaksije ili čak i u drugim galaksijama.

Visokoenergetske gama zrake se također mogu proizvesti raspadadom čestica, možda i raspadom čestica tamne tvari, koje su ostaci vjerojatno još iz razdoblja velikoga praska. Dakle, gama zrake nam mogu donijeti uvid u prirodu i konzistenciju tamne tvari.

1.4. CTA sklopovlje

Trenutačna mreža Čerenkovljevih teleskopa koristi najviše 4 teleskopa, koji pružaju stereo sliku pljuska čestica, gdje se najveći dio sekundarnog pljuska čestica gleda sa samo dva ili tri teleskopa. Nova mreža (slika 3), koja se sastoji od nekoliko desetaka teleskopa, omogućit će detekciju gama zraka nad većim područjem, što će omogućiti dramatično poboljšanu detekciju gama zraka, a u isto vrijeme pružiti i bolji način pregleda određenog elementa u pojedinom pljusku čestica. Ovo će rezultirati boljom kutnom rezolucijom i boljom separacijom pozadinskih događanja uzrokovanih kozmičkim zračenjem.

Predviđena su dva mjesta s CTA teleskopima, na sjevernoj i južnoj zemljinoj polutci, kako bi se pokrilo čitavo nebo.

U mogućem dizajnu na južnoj polutci, CTA će se sastojati od tri tipa teleskopa s različitom veličinom zrcala, kako bi mogli pokriti čitavi energetski spektar, a na sjevernoj polutci bi se CTA sastojao od dva veća tipa teleskopa.

Slika 3: Mreža teleskopa CTA.

Niskoenergetski niz teleskopa (između 20 i 200 GeV) sastojat će se od nekoliko (vjerojatno 4) teleskopa 23-metarskog promjera s osrednjim područjem pregleda, veličine od otprilike 4.5 stupnjeva.

Srednjeenergetski niz teleskopa (između 100 GeV i 10 TeV) sastojat će se od oko 40 teleskopa od 12 metara promjera s područjem pregleda od oko 7 stupnjeva.

Visokoenergetski niz teleskopa (nekoliko TeV do 300 TeV) sastojati će se od oko 70 malih, 4-metarskih, teleskopa s područjem pregleda od oko 9.1 do 9.6 stupnjeva.

1.5. MAGIC teleskopi

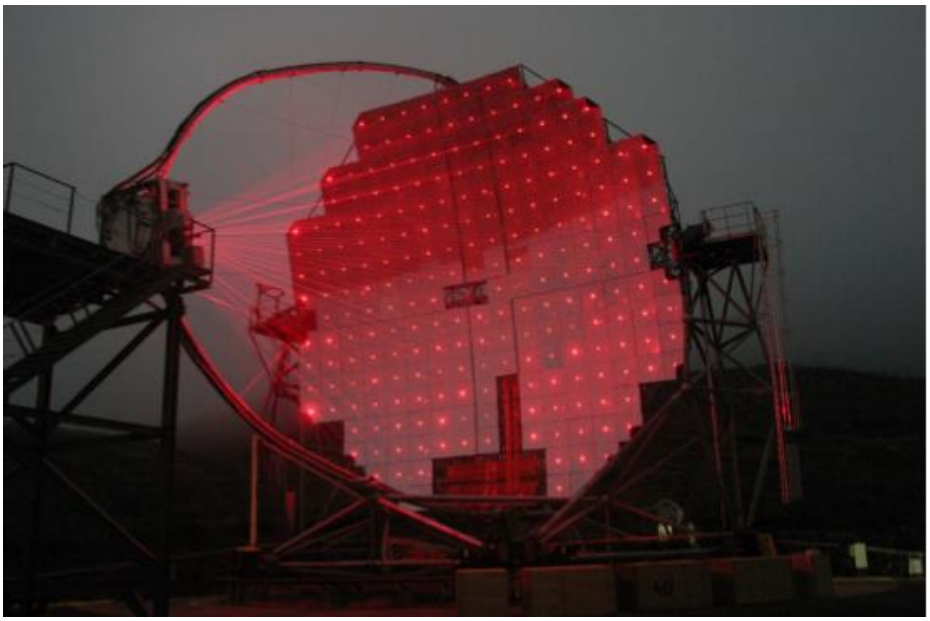
MAGIC teleskopi (Slika 4) [3] , najveći su sadašnji Čerenkovljevi teleskopi čiji reflektori imaju promjer od 17 metara i trenutno su jedni od znanstveno najproduktivnijih instrumenata na području astročestične fizike. MAGIC I MAGIC II nalaze se na vrhu otoka La Palme (Kanarsko otočje), unutar opservatorija Roque de los Muchachos. Nalaze se na nadmorskoj visini od 2200 m. Čerenokovljevi teleskopi na ovoj lokaciji optimalno rade. Relativna vlažnost zraka je pretežito ispod je 10% i zrak ima vrlo povoljan sastav za promatranje Čerenkovljeve svijetlosti valne duljine 290-700 nm. Također je bitno da je oblačnih dana manje od 15% kroz godinu, što je bio jedan bitan faktor za odabir lokacije.

1.5.1. Struktura MAGIC teleskopa

Dva glavna tehnička zahtjeva za promatranja teleskopima MAGIC su nizak prag energije koji teleskop može detektirati i kratko vrijeme reagiranja. Dizajn teleskopa fokusiran je na izradu teleskopa sa što većom površinom reflektora, tako promjer reflektora od 17 metara omogućuje detekciju gama zraka energije od oko 30 GeV i više, dok svi ostali IACT teleskopi (HESS, VERITAS) imaju donji prag detekcije iznad 100 GeV-a. Teleskop je konstruiran od štapova od karbonskih vlakana, pa je ukupna masa teleskopa relativno niska i iznosi 40 tona. Tako se dobio brži i pokretniji teleskop. Teleskope pomiču/usmjeravaju dva motora, jedan duž azimutnog kuta, a drugi duž zenitnog kuta. Reflektor je načinjen od aluminijskih ploča koje su lakše od standardnih staklenih površina, a poredane tako da formiraju paraboličnu površinu. Pri konstrukciji mnogo se pažnje posvetilo dizajnu da bi se dobio što manji okretni moment strukture, a da bi ona bila dovoljno čvrsta i postojana za sve vremenske uvjete. Ukupna masa reflektivne strukture zajedno sa 1000 aluminijskih ogledala je 9 tona i može podnijeti vjetar brzine više od 165 m/s sa minimalnom deformacijom od 3,5 cm. Razvijena je aktivna kontrola položaja zrcala (AMC), radi finog podešavanja svakog pojedinog zrcala kako bi se postigao što bolji fokus reflektora. Zrcala nisu direktno instalirana na karbonsku nosivu strukturu, već se nalaze na posebnim panelima u grupi od četiri. Svaki panel ima motor koji služi za dovođenje zrcala u odgovarajući fokus. U samom središtu reflektora se nalazi CCD kamera koja mjeri u koji dio kamere upada reflektirana testna laserska zraka s pojedinog zrcala što se može vidjeti na slici (slika 5).

Slika 4: MAGIC teleskopi

Zrcala se pomiču posebnim programom koji upravlja motorima panela, te se tako mijenja fokus pojedinog zrcala. Nakon pomicanja treba provjeriti slike lasera u kameri, te se zatim prelazi na sljedeće zrcalo i tako redom do zadnjega. Kompletan proces fokusiranja cijelog reflektora traje oko 5 minuta i mora se napraviti svaki put kad je znatna varijacija u zenitnom kutu promatranog izvora.



Slika 5: Lasersko fokusiranje zrcala.

Kamera koja se nalazi u fokusu reflektora pretvara Čerenkovljev bljesak svjetlosti u električni naboj koji se dalje elektronički obrađuje, digitalizira i zapisuje u memoriju računala za daljnju obradu. Kamera uzima sliku Čerenkovljevog bljeska iz koje se zaključuje je li bljesak izazvan gama zrakom ili pozadinom (npr. nabijenom kozmičkom zrakom). Slika u kameri slična slici izazvanoj gama zrakom, može biti i rezultat fluktuacije svjetlosne pozadine noćnog neba ili naprosto rezultat različitih svjetlosnih smetnji iz umjetnih izvora. Osjetljivost kamere zajedno sa zrcalima igra odlučujuću ulogu u snižavanju energijskog praga teleskopa Eth (energy threshold). Eth je obrnuto proporcionalan efikasnosti detekcije Čerenkovljvih fotona, dakle veća osjetljivost kamere znači niži Eth teleskopa. MAGIC teleskopi postižu nizak Eth zahvaljujući velikoj površini zrcala, upotrebi foto-multiplikatorskih (PMT) cijevi s visokom kvantnom efikasnošću, te redukcijom mrtvih područja svjetlosnim koncentratorima.

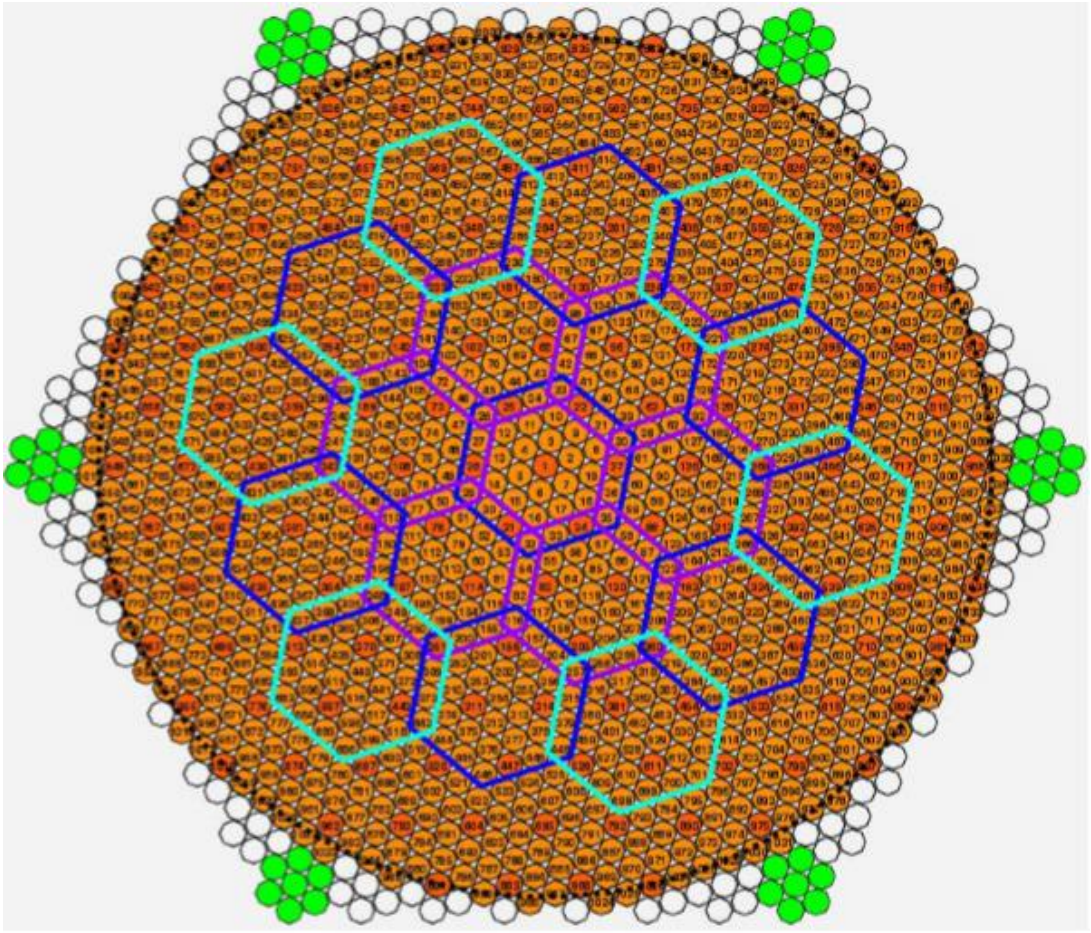
1.5.2. MAGIC I i MAGIC II

MAGIC 1 teleskop je dovršen krajem 2003. godine, dok je MAGIC 2 teleskop počeo s radom 2009. godine.

MAGIC I teleskop je posljednji put unaprijeđen 2012. godine [4]. Glavni cilj ovoga unaprjeđenja je bio zamijeniti staru kameru s novom, također i Level I okidač (eng. trigger). Jednako tako, promijenjen je način na koji se iščitavaju podaci (eng. readout) cijelog sustava. Readout se sastoji od tehnologije bazirane na Drs4 čipovima. Ploča koja prima signale je unaprijeđena da bude ista kao i u MAGIC II teleskopa. Postavljanje je prikazano na slici (slika 6).

Slika 6: Postavljanje nove kamere teleskopa MAGIC I.

Kamera za MAGIC II sastoji se od 1183 piksela od kojih su 1039 trenutno u upotrebi, a dodano je i 42 nova piksela (HPDs) koji su trenutno u fazi testiranja. Pikseli su grupirani u heksagonalnu strukturu od 7 piksela formirajući tako jedan klaster, koji se može jednostavno skinuti i zamijeniti. Pikseli su također napravljeni od PMT cijevi. Kamera ukupno ima 169 klastera od kojih 92 centralnih klastera čini okidačko (tigger) područje, pokrivajući tako 2.5° Field of view (FoV). Kamera (Slika 7) je postavljena u fokus reflektora udaljena od njega 17 metara. Sve mehaničke i elektroničke komponente na kameri moraju biti minimalne mase, pa je većina napravljena od aluminija tako da je ukupna masa kamere 600 kg. Elektronika kamere napajana je sa dva izvora od 5V smještenih izvan kućišta kamere. Centralni dio tijela kamere sastoji se od dva hladnjaka (cooling plate), odnosno dvije ploče u kojima se nalazi rashladna tekućina radi stabilizacije temperature elektronike. Rashladni sistem je dizajniran da radi na temperaturi od -10 do +30°C i njegova maksimalna potrošnja električne energije je 8 kW dok elektronika kamere troši manje od 1 kW. U samim kutovima kamere nalazi se 6 klastera koji ne sadrže niti jednu PMT cijev, već su predviđeni za testiranje novih tipova fotodetektora. Trenutno su dodana nova 42 hibridna fotodetektora u kameru i upravo se testiranju. Svi ostali elektronički senzori i kontrole smješteni su u kućištima s druge strane hladnjaka (cooling plate).

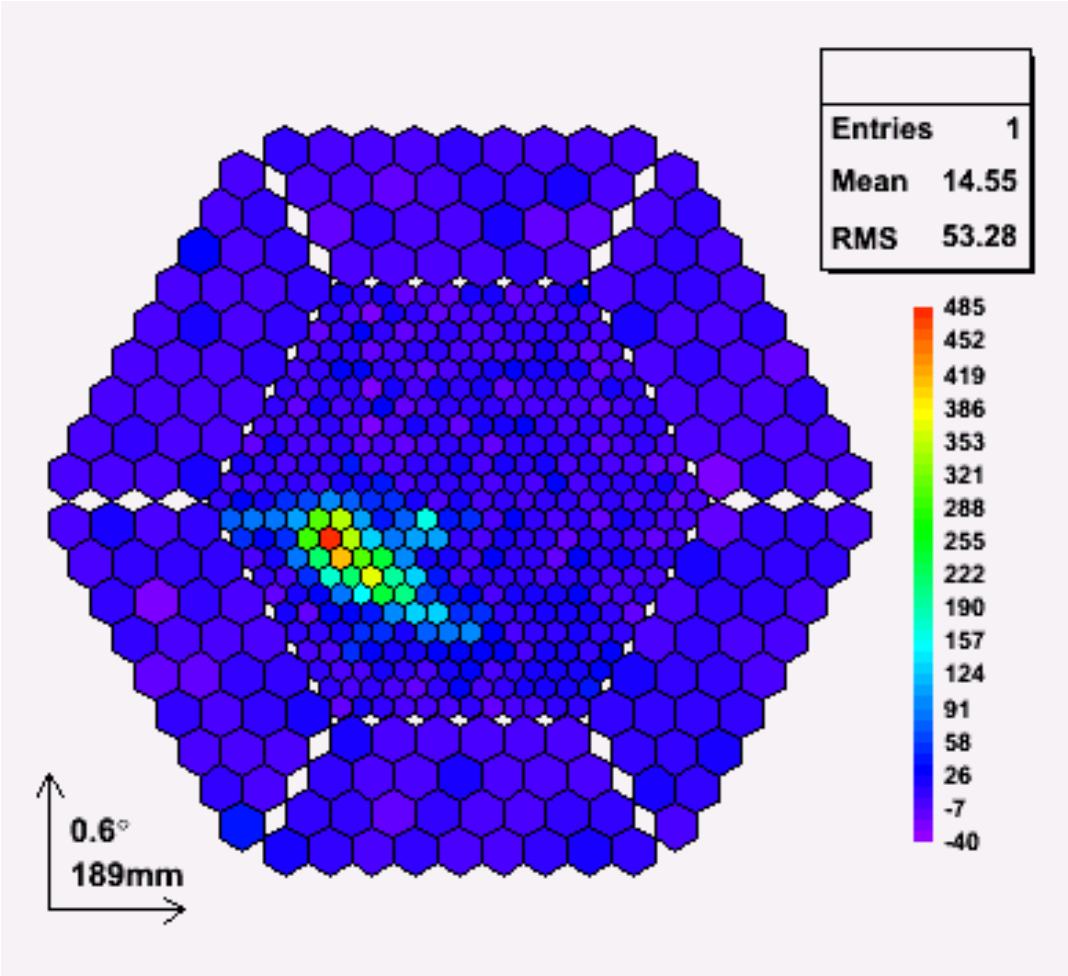


Slika 7: Pikseli kamere teleskopa MAGIC II.

PMT cijevi se sastoje od polupropusne fotokatode i 6 dinoda prije anode. Svaka cijevi sastoji se od elektroničke opreme (HV) koja služi za napajanje i pojačavanje signala PMT-a. Napon na katodi i dinodama generiran je Cockroft-Walton DC-DC konverterom koji može generirati napon veći od 1250 V. Signal iz pojedine PMT cijevi je pojačan za približno 25 dB AC predpojačalom pojasne širine 700 MHz. Predpojačalo služi i za zaštitu od potencijalno destruktivnog napona iz PMT cijevi. Prethodno pojačan PMT signal se pretvara u optički signal pomoću VCSEL-a koji se prenosi kroz optički kabel duljine 162 metra do kontrolne sobe, gdje se optički signal ponovo pretvara u električni. Analogno optička pretvorba vrši se elektronikom smještenom u samom kućištu pojedinog klastera, koje je spojena s hladnjacima da bi se minimizirala promjena temperature. Vrijednosti struja, napona i temperatura svake pojedine PMT cijevi, jednako kao i sustava za pretvorbu električnog u optički signal, se konstantno zapisuju tijekom opažanja, što je vrlo važno da bi se znalo da li kamera radi ispravno. SCC (Slow control of the camera) elektronički sklopovi, instalirani na svakom klasteru kontroliraju

stanje kamere mjereći različite parametre kao što su HV (visoki napon) pojedinog piksela, struja i temperatura PMT-a, napajanje i temperatura VCSEL, itd. SCC također upravlja poklopcima kamere (lids) radi zaštite kamere od prejakog svjetla, kiše, ili jakog vjetra.

Na slici (slika 8) je prikazana detekcija pljuska nastalog od gama zrake (gama shower) u prvotnoj verziji kamere MAGIC I teleskopa.



Slika 8: Gama shower

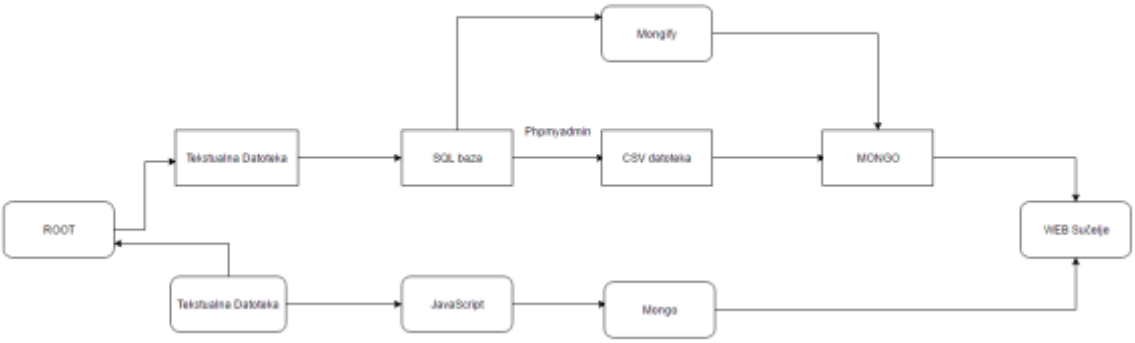
1.5.3. Okidači

Okidači (eng. trigger) donose odluku hoće li određeni događaj biti prihvaćen i kasnije analiziran. Služe za eliminaciju lažnih svjetlosnih signala kao što su pozadinski sjaj noćnog neba ili razni drugih izvora pozadine. Sustav okidača se nalazi u elektroničkoj sobi i dio je sustava za prikupljanje podataka. Postoje tri razine triggera u svakom MAGIC teleskopu. Prva razina (L1T) je diskriminator koji signal iz pojedinog piksela šalje u dalju obradu ako je signal

iznad praga diskriminatora. Druga razina (L2T) je složenija, provjerava je su li signali koji su prošli prvu tigger razinu iz 2, 3 ili 4 susjedna pixela, ovisno o tome koja se tiggerska logika koristi za pojedino mjerenje. Treća triggerska razina (L3T) ili stereo trigger prihvaća događaj iz oba teleskopa samo ako su detektirani unutar vremenskog prozora od 100 nanosekundi, kako bi se osiguralo da su teleskopi, a koji su međusobno udaljeni 75 metara, zaista zabilježili isti Čerenkovljev bljesak, tj. istu gama zraku.

2. Zahtjevi i rješenja

U ovom poglavlju prikazati ćemo niz zadataka koji su postavljeni timu za izradu sustava za dugotrajnu kontrolu kvalitete podataka MAGIC teleskopa. Dijagram toka možemo vidjeti na slici (slika 9)



Slika 9: Dijagram toka

Prvi zadatak bio je prebaciti postojeću MySql bazu podataka na MongoDB sustav. Drugi je zadatak bio spremiti podatke koje teleskop MAGIC proizvodi za dugotrajnu kontrolu kvalitete podataka direktno u novu MongoDB bazu podataka. Treći zadatak je bio izrada web sučelja u kojem će se, korištenjem alata za iscrtavanje, prikazivati podatke koje prikuplja teleskop MAGIC. Zadnji zadatak je bila integracija cijelog sustava u jednu cjelinu.

2.1. Prebacivanje podataka iz postojeće baze MySql u MongoDB

2.1.1. Mongify

Za prebacivanje trenutne baze podataka u MongoDB možemo koristiti dodatak za Ruby koji se naziva Mongify [5]. Radi se o alatu za prebacivanje podataka iz Sql baze u MongoDB, a pomaže da prebacimo podatke iz jedne baze u drugu bez straha da će se poremetiti primarni i strani ključevi. Omogućuje da spremimo podatke kao dokument, uključujući i veze između njih, te sve navedeno obavlja u par jednostavnih koraka. Mongify je najlakše koristiti u Linuxu

iz terminala. Sve što treba napraviti za instalaciju je unijeti naredbu `gem install mongify`. Nakon što se modul instalira dostupne su kratke upute za korištenje, kao što je prikazano na slici (Slika 10).

```
Usage: mongify command database.config [database_translation.rb]

Commands:
  "check" or "ck"           >> Checks connection for sql and no_sql databases
[configuration_file]
  "process" or "pr"         >> Takes a translation and process it to mongodb [configuration_file,
 translation_file]
  "translation" or "tr"     >> Outputs a translation file from a sql connection [configuration_fi
le]

Examples:

  mongify translation database.config
  mongify tr database.config
  mongify check database.config
  mongify process database.config database_translation.rb

 Common options:
    -h, --help                      Show this message
    -v, --version                   Show version
```

Slika 10: Kratke upute za Mongify.

Kako bi se izvršila pretvorba baze, treba napraviti dvije stvari: postaviti točne podatke unutar konfiguracijskog dokumenta, kao što su tip baze podataka koji koristimo, te podaci za spajanje na nju, korisničko ime i lozinka. Ovaj korak se koristi kako bi se locirale relacije u SQL bazi koje će se naknadno prebaciti u MongoDB bazu. Drugi korak je postaviti translacijski dokument koji će se koristiti za prevođenje SQL baze prije spremanja u MongoDB sustav.

Stoga, prije izvršavanja pretvorbe, potrebno je postaviti navedenu konfiguracijsku datoteku. Primjer toga je prikazan u nastavku, na slici (Slika 11).

```
sql_connection do
  adapter    "mysql"
  host       "localhost"
  username   "root"
  password   "passw0rd"
  database   "my_database"
end

mongodb_connection do
  host       "localhost"
  database   "my_database"
end
```

Slika 11: Primjer konfiguracijske datoteke.

Kako bi Mongify znao što napraviti s zadanim podacima, potrebno je zadati translacijski dokument: `translation.rb`, (slika 12), kao što je prikazano.

```
table "users" do
  column "id", :key
  column "first_name", :string
  column "last_name", :string
  column "created_at", :datetime
  column "updated_at", :datetime
end

table "posts" do
  column "id", :key
  column "title", :string
  column "owner_id", :integer, :references => :users
  column "body", :text
  column "published_at", :datetime
  column "created_at", :datetime
  column "updated_at", :datetime
end

table "comments", :embed_in => :posts, :on => :post_id do
  column "id", :key
  column "body", :text
  column "post_id", :integer, :referneces => :posts
  column "user_id", :integer, :references => :users
  column "created_at", :datetime
  column "updated_at", :datetime
end

table "preferences", :embed_in => :users, :as => :object do
  column "id", :key
  column "user_id", :integer, :references => "users"
  column "notify_by_email", :boolean
end

table "notes", :embed_in => true, :polymorphic => 'notable' do
  column "id", :key
  column "user_id", :integer, :references => "users"
  column "notable_id", :integer
  column "notable_type", :string
  column "body", :text
  column "created_at", :datetime
  column "updated_at", :datetime
end
```

Slika 12: Translacijski dokument.

Naredbe koje se koriste u Mongify alatu su sljedeće:

- `check` - naredba koja provjerava je li database.config dokument valjan i jesu li povezanosti dobro postavljene. Primjer korištenja:

```
mongify check database.config
```

- `translation`. Translacija se koristi za automatsko generiranje translacije iz Sql baze podataka. Primjer korištenja je idući:

```
mongify translation database.config
```

  Ako se želi ispisati rezultat u datoteku `translation.rb`, koristi se:

```
mongify translation database.config > translation.rb
```

- `process` - Jednom kada se zada valjani translacijski dokument, u Mongify alatu se može prebaciti podatke iz SQL u mongo koristeći naredbu `process`, kao što je prikazano u nastavku:

```
mongify process database.config translation.rb
```

Na kraju, za provjeru detalja nastale mongo baze, potrebno je u mongo okruženju izvršiti naredbu

```
db.status()
```

U okviru ovog rada istražen je dodatni, nešto jednostavniji, način pretvaranja SQL u mongo format, koristeći Phpmyadmin alat, što je opisano u idućem odjeljku.

2.1.2. Phpmyadmin

Phpmyadmin je besplatni program napisan u php jeziku, koji je namijenjen za administriranje MySql baza podataka na nekome web servisu ili nekoj web stranici. Phpmyadmin podržava razne operacije u MySql-u. Neke od njih su spajanje više baza podataka, tablica, redaka, postavljanje relacija, indeksiranje, dodavanje ovlasti itd.

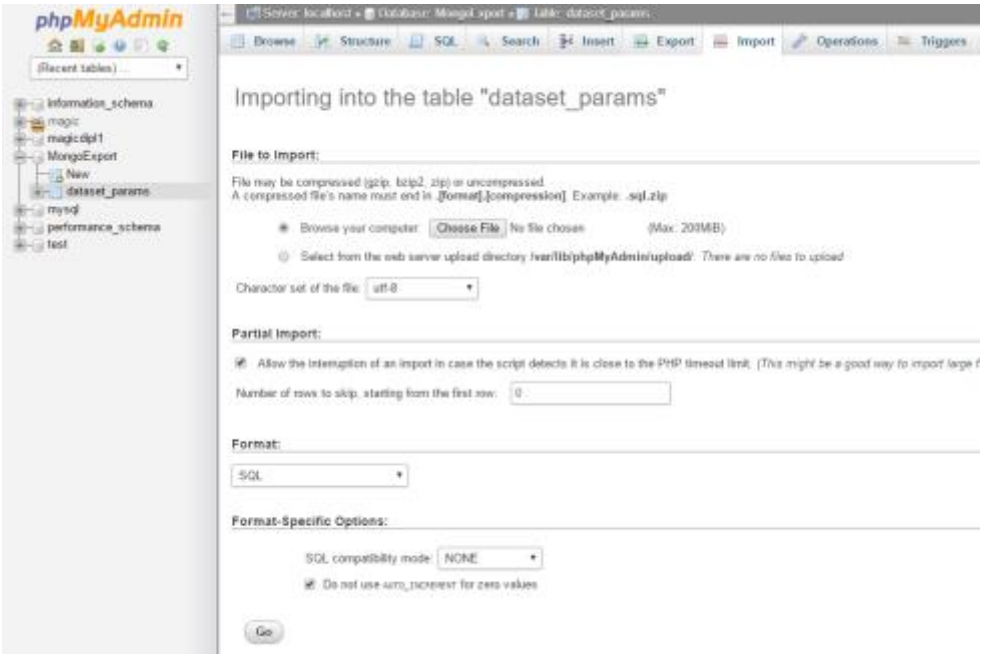
Popis mogućnosti priložen je u dokumentaciji:

- Intuitivno web sučelje
- Podržava većinu MySql mogućnosti kao:
    - Pregledavanje i brisanje baze podataka, njenih tablica, relacija i indeks-a (primarnih i sekundarnih ključeva)
    - Stvaranje, kopiranje, brisanje, preimenovanje i mijenjanje tablica, polja i indeks polja
    - Održavanje servera i bazi podataka

    - Izvršavanje raznih SQL upita
    - Postavljane raznih SQL privilegija što je veoma važno dobro postaviti kako bi se zaštitili od SQL napada
- Učitavanje baze podataka iz **SQL ili CSV** dokumenta
- Zapisivanje baze u različitim formatima: **CSV, SQL, Xml, Pdf** i mnogi drugi
- Administriranje raznih servera
- Mogućnosti grafičkog prikaza baze i relacije među njima

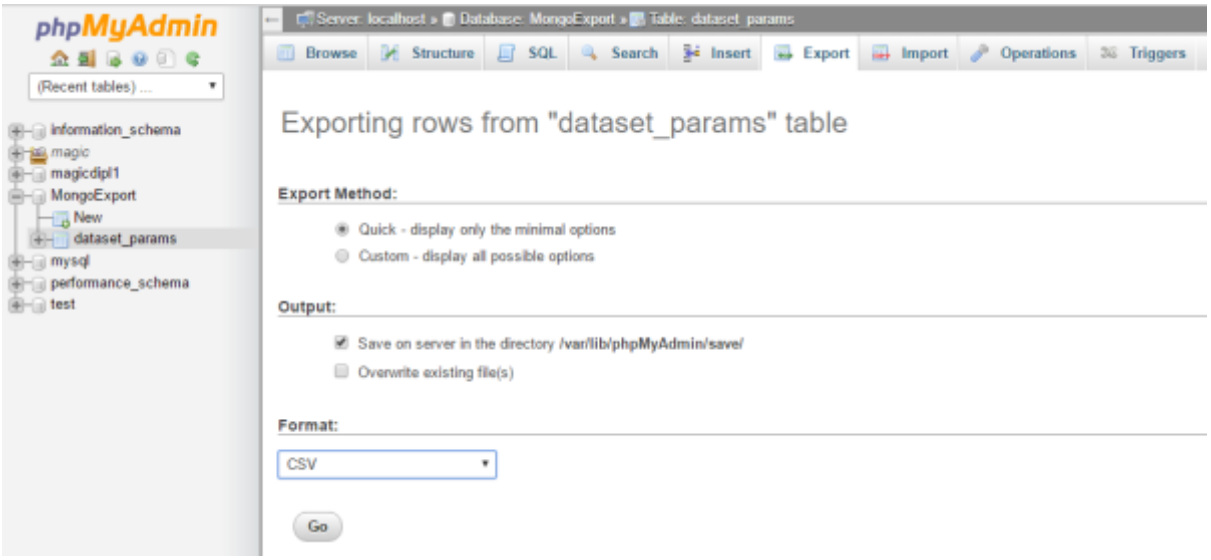
Nama najbitnija stavka ovog navedenog popisa je prebacivanje stare baze u SQL formatu u CSV dokument, kako bi naknadno mogli učitati CSV dokument u novu mongo bazu.

Nakon instaliranja phpmyadmin-a u datoteci php.ini potrebno je podesiti gornju granicu za veličinu ulazne datoteke, max_upload_file_size, koja je u startu postavljena na 8 Mb, a pošto naš backup baze ima veličinu veću od ove vrijednosti (postavljeno na 800 Mb). Potrebno je resetirati server kako bi se promjene izvršile. U phpmyadmin-u je, potom, potrebno kreirati novu bazu u koju ćemo učitati naš backup (slika 13).



Slika 13: Učitavanje sql baze u phpmyadmin

Kada se postojeća SQL baza učita, klikom na tab *export* odabere se željena vrsta izlazne datoteke, u našem slučaju to je CSV, te odluči da li će se datoteka spremiti direktno u direktorij na serveru ili se želimo preuzeti (download), slika 14.



Slika 14: Kreiranje CSV datoteke

Kad dobijemo željenu datoteku, potrebno je logirati se na server i doći do mjesta gdje se nalazi CSV datoteka. Potrebno je ući u mongo strukturu, naredbom *mongo*, te zadati naredbu:

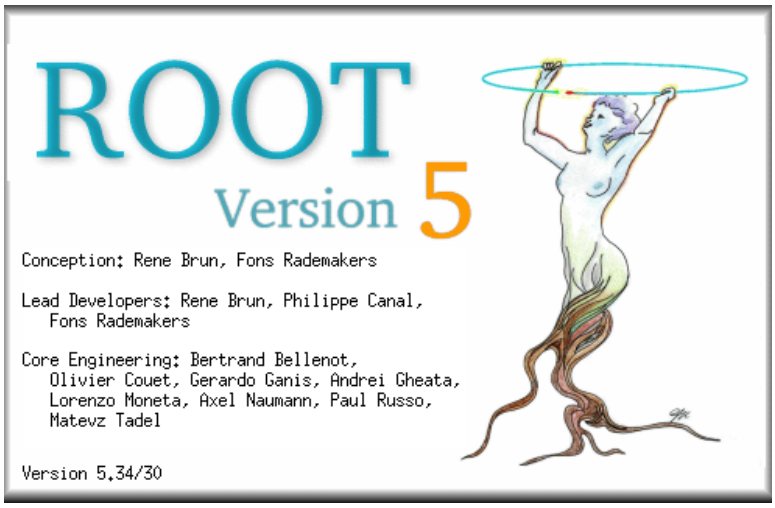
```
mongoimport -d symmetrybaza -c things --type csv --file
mysql_exported_table-dataset_params.csv –headerline
```

Ova naredba stvara novu bazu pod imenom symmetrybaza u mongu i u njoj napravi kolekciju pod imenom things, zatim napuni kolekciju podacima koji su zapisani u mysql_exported_table-dataset_params.csv.

2.2. Punjenje MongoDB podacima iz tekstualnih datoteka

Jedan od mojih zadataka je bio i spremanje podatka, koje teleskop MAGIC proizvodi za dugotrajnu kontrolu kvalitete podataka, direktno u novu MongoDB bazu podataka. Znatan dio podataka koje proizvode MAGIC teleskopi se pohranjuje u ROOT formatima. ROOT (Slika 15) je programska biblioteka (eng. *library*) koja je nastala na CERN-u, a nudi brojne mogućnosti, kao što su alati za obradu i spremanje podataka, iscrtavanje grafova i histograma,

itd. Na dnevnoj bazi se rezultati dobiveni praćenjem rada MAGIC teleskopa zapisuju u ROOT formatu. Podaci iz ROOT datoteka (primjer korištenih datoteka za dan 01.07.2016. je CheckDomino_M1_2016_07_01.root i CC_M1_2016_07_01.root), koristeći poseban ROOT macro, se odabiru i ispisuju u tekstualnu datoteku koja izgleda kao na slici (Slika 16). Na slici se jasno vidi da je prvi red tekstualne datoteke zapravo datum, naveden u root datotekama, i označava dan kad su podaci zabilježeni.



Slika 15: ROOT.

Prvi stupac u datoteci je ime kolekcije, drugi stupac sadrži naziv pojedinih vrijednosti (min, max, mean, median, rms) kolekcije i treći stupac sadrži same vrijednosti.

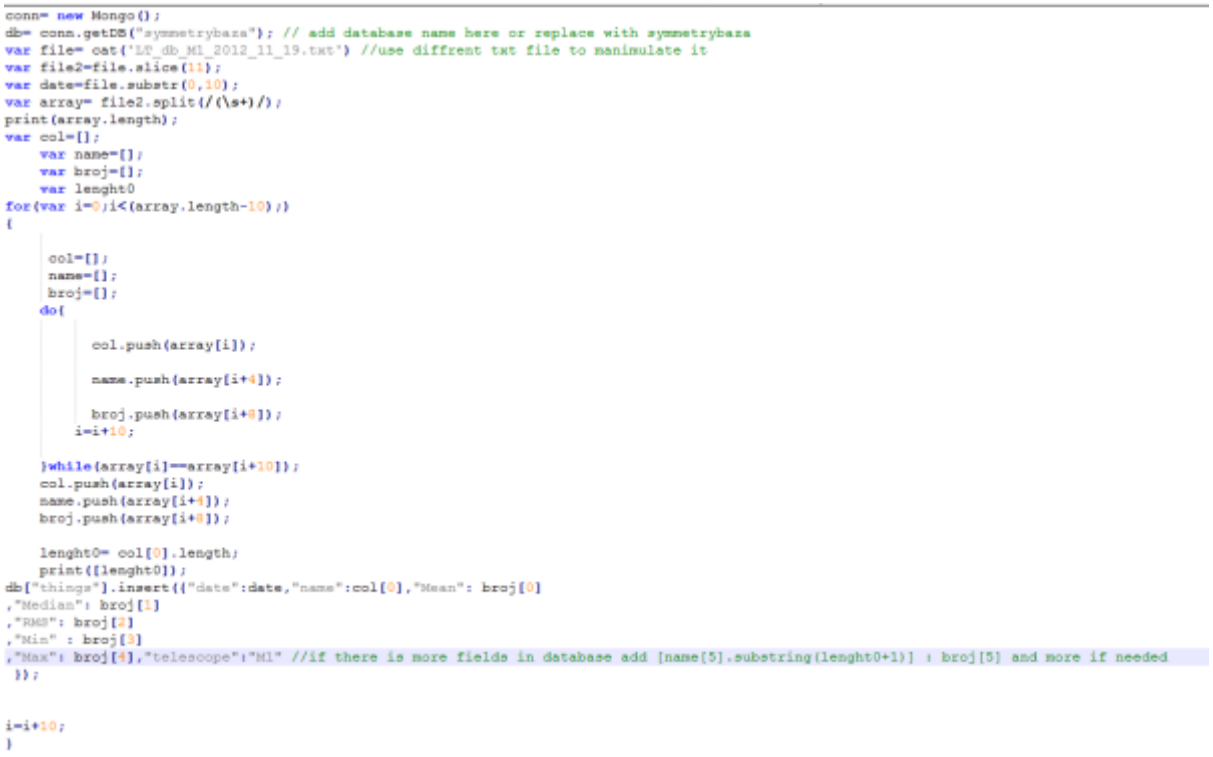
```
2012-11-19
CalQ_Cal , CalQ_Cal_Mean   , 5837.730676
CalQ_Cal , CalQ_Cal_Median , 5987.613363
CalQ_Cal , CalQ_Cal_RMS    , 1177.853848
CalQ_Cal , CalQ_Cal_Min    , 19.033334
CalQ_Cal , CalQ_Cal_Max    , 8596.906641
CalQ_Int , CalQ_Int_Mean   , 8272.950825
CalQ_Int , CalQ_Int_Median , 8581.994873
CalQ_Int , CalQ_Int_RMS    , 1614.734683
CalQ_Int , CalQ_Int_Min    , 69.008480
CalQ_Int , CalQ_Int_Max    , 9497.246441
CalQ_Sig , CalQ_Sig_Mean   , 4279.558451
CalQ_Sig , CalQ_Sig_Median , 4296.838515
CalQ_Sig , CalQ_Sig_RMS    , 825.446594
CalQ_Sig , CalQ_Sig_Min    , 2.827739
CalQ_Sig , CalQ_Sig_Max    , 6071.036922
Bias_Sig , Bias_Sig_Mean   , 217.258015
Bias_Sig , Bias_Sig_Median , 220.883216
Bias_Sig , Bias_Sig_RMS    , 32.484006
Bias_Sig , Bias_Sig_Min    , 62.596820
Bias_Sig , Bias_Sig_Max    , 278.431978
HitFrac_Sig , HitFrac_Sig_Mean   , 0.420563
HitFrac_Sig , HitFrac_Sig_Median , 0.411272
HitFrac_Sig , HitFrac_Sig_RMS    , 0.144425
HitFrac_Sig , HitFrac_Sig_Min    , 0.000018
HitFrac_Sig , HitFrac_Sig_Max    , 0.838569
ArrTm_Cal , ArrTm_Cal_Mean   , 30.268675
ArrTm_Cal , ArrTm_Cal_Median , 30.145000
ArrTm_Cal , ArrTm_Cal_RMS    , 2.836786
ArrTm_Cal , ArrTm_Cal_Min    , 22.407334
ArrTm_Cal , ArrTm_Cal_Max    , 38.205333
ArrTm_Int , ArrTm_Int_Mean   , 28.980680
ArrTm_Int , ArrTm_Int_Median , 28.857067
ArrTm_Int , ArrTm_Int_RMS    , 2.877088
ArrTm_Int , ArrTm_Int_Min    , 21.315035
ArrTm_Int , ArrTm_Int_Max    , 37.518003
ArrTm_Sig , ArrTm_Sig_Mean   , 21.263435
ArrTm_Sig , ArrTm_Sig_Median , 21.578984
```

Slika 16: Tekstualna datoteka s podacima.

Macro kojim se puni tekstualna datoteka je napisan u C++-u, koristeći ROOT, te spomenute ulazne ROOT datoteke. Kako bi se macro izvršio, potrebno je koristiti i MAGIC softverski paket, MARS, koji omogućuje korištenje svih potrebnih objekata. Macro se pokreće idućom naredbom, i kao što je navedeno, proizvodi izlaznu tekstualnu datoteku.

```
root.exe –b –q 'LongTerm_db(“data/CheckDomino_M1_2016_07_01.root“,
“data/CC_M1_ 2016_07_01_datacheck.root“)'
```

Nakon toga, izradio sam JavaScript program kojim se podaci iz tekstualne datoteke zapisuju u MongoDB bazu podataka. To je napravljeno na način da se u programu otvori tekstualna datoteka, te prođe kroz nju koristeći *for* petlju. Putem se gleda gdje koji podatak ide u bazu, te ih se sve raspodijeli. Programski kod prikazan je na slici (Slika 17).



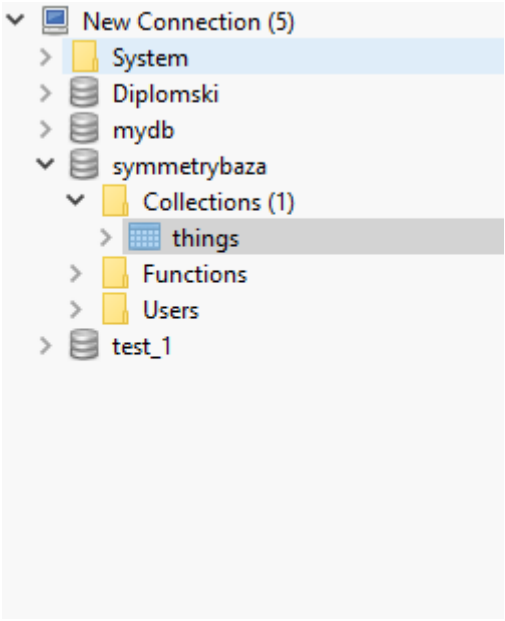
```
conn= new Mongo();
db= conn.getDB("symmetrybaza"); // add database name here or replace with symmetrybaza
var file= cat('LT_db_M1_2012_11_19.txt') //use diffrent txt file to manimulate it
var file2=file.slice(11);
var date=file.substr(0,10);
var array= file2.split(/(\s+)/);
print(array.length);
var col=[];
    var name=[];
    var broj=[];
    var lenght0
for(var i=0;i<(array.length-10);)
{

     col=[];
     name=[];
     broj=[];
    do{

         col.push(array[i]);

         name.push(array[i+4]);

         broj.push(array[i+8]);
        i=i+10;

    }while(array[i]==array[i+10]);
    col.push(array[i]);
    name.push(array[i+4]);
    broj.push(array[i+8]);

    lenght0= col[0].length;
    print([lenght0]);
db["things"].insert({"date":date,"name":col[0],"Mean": broj[0]
,"Median": broj[1]
,"RMS": broj[2]
,"Min" : broj[3]
,"Max": broj[4],"telescope":"M1" //if there is more fields in database add [name[5].substring(lenght0+1)] : broj[5] and more if needed
 });


i=i+10;
}
```

Slika 17: JavaScript program za prebacivanje podataka iz tekstualne datoteke u MongoDB.

Program je potrebno izvršiti unutar MongoDB sustava korištenjem sljedeće naredbe:

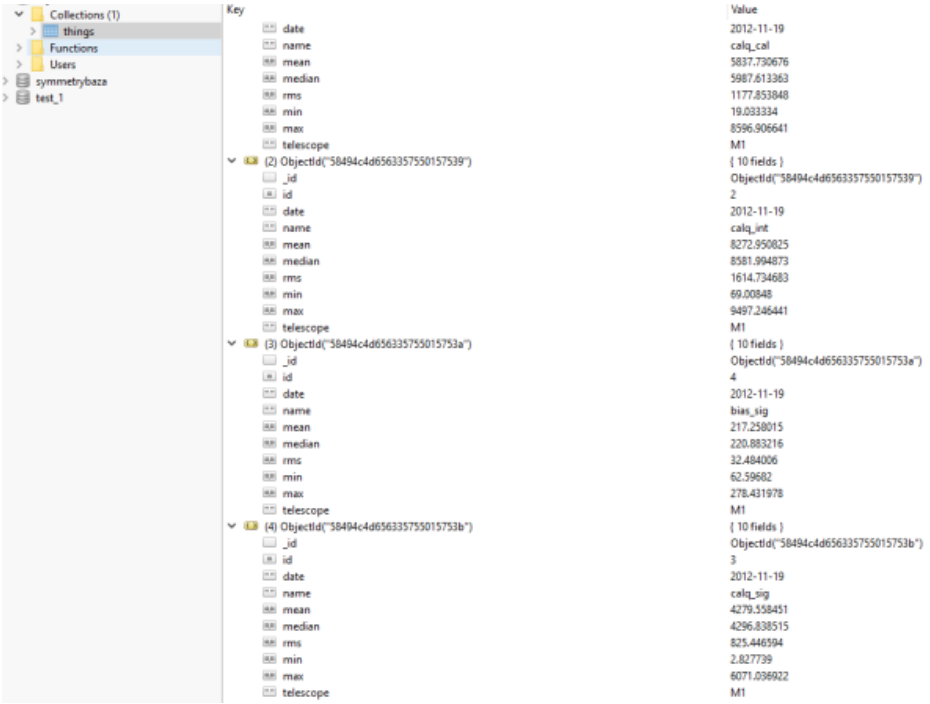
```
load ('Prebacivanje.js')
```

Koristimo naziv `Prebacivanje`.js budući da je to naziv datoteke unutar koje se nalazi navedeni programski kod. Nakon izvršavanja programa, možemo primijetiti da se unutar MongoDB sustava nalazi naša baza podataka, *symmetrybaza*, (Slika 18).



Slika 18: Baza podataka

Sve podaci iz tekstualne datoteke su pohranjeni u mongo bazi (Slika 19).



Slika 19: Prikaz polja s podacima u mongo bazi.

2.3. Izrada web sučelja za prikaz podatak teleskopa MAGIC

Jedan od planiranih zadataka bio je i izrada web sučelja za prikazivanje dugotrajnih, višednevnih, podataka za kontrolu kvalitete MAGIC teleskopa. Korištene su različite tehnologije. Pyramid, web okruženje otvorenog koda (eng. *open source*) koje je napisano u Pythonu i temeljeno na WSGI (eng. *Web Server Gateway Interface*) specifikaciji. Pyramid je minimalistički, objektno orijentiran MVC (eng. *Model View Controler*) razvojni okvir (eng. *framework*). Neovisan je o platformi i vrsti baze podataka kojoj pristupa, tako da uz brojne druge, ima i podršku za MongoDb sustav. Pyramid se koristi u okviru programskog jezika Python, te je potrebno napraviti virtualno okruženje u kojem će se alat koristiti. Na slici (Slika 20) je prikazan primjer postavljanja virtualnog okruženja.

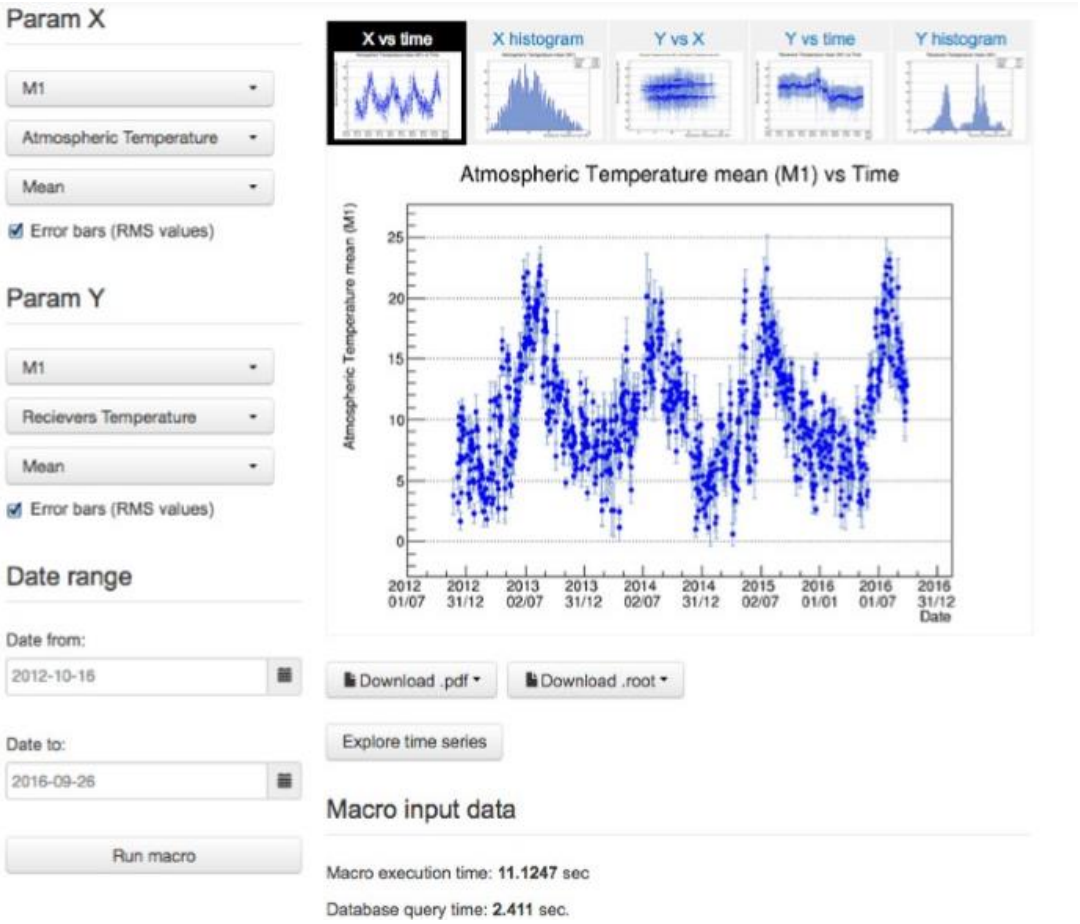
```
C:\Users\Ante>md virtualenv

C:\Users\Ante>Venv = C:\virtualenv
'Venv' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Ante>set Venv = C:\virtualenv

C:\Users\Ante>cd ..

C:\Users>cd ..

C:\>cd Python27

C:\Python27>python -m virtualenv %VENV$
```

Slika 20: Primjer postavljanja Python virtualnog okruženja.

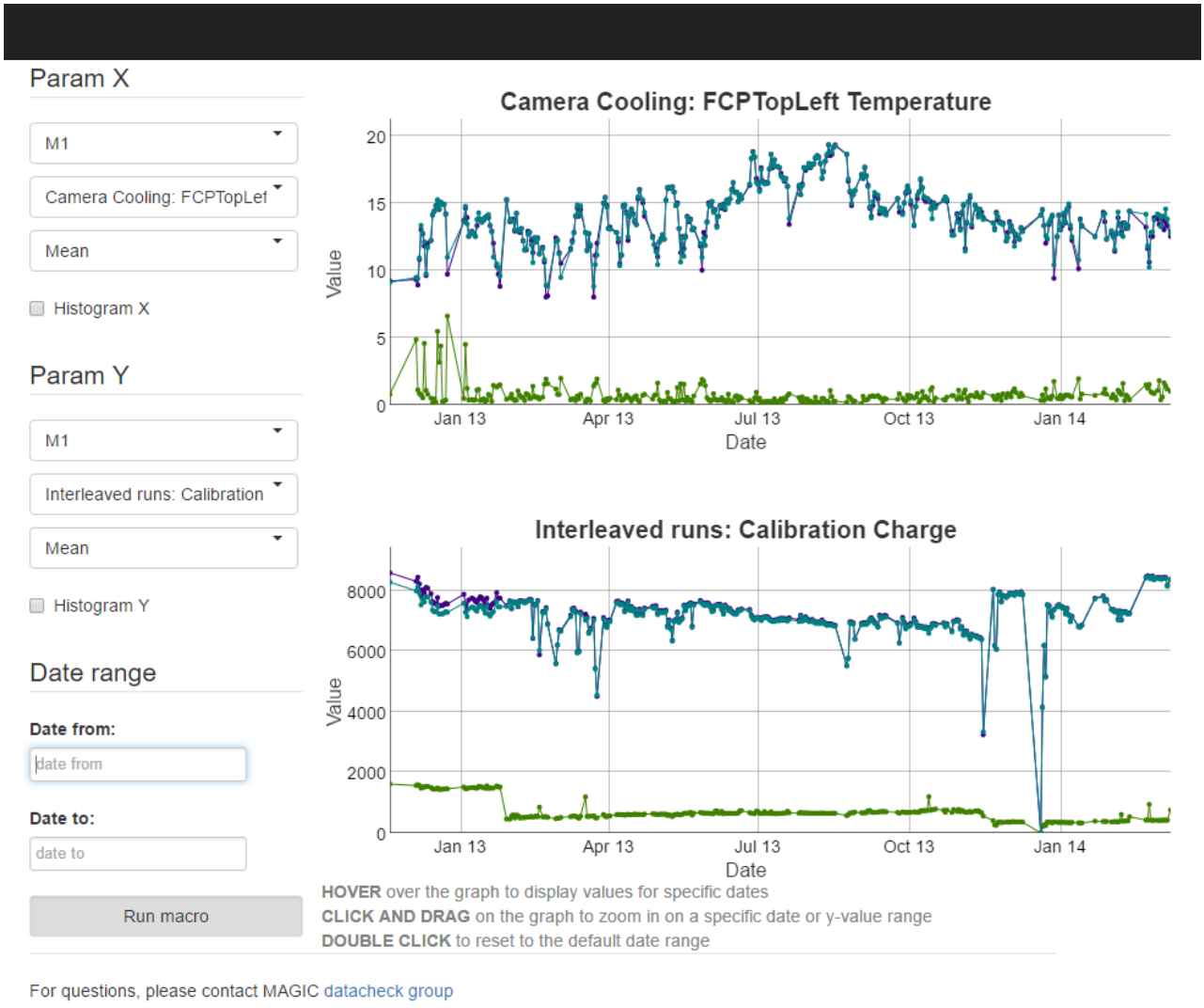
Nakon instalacije Pyramida potrebno je izraditi programsku aplikaciju web sučelja. U Pyramidu se pristupa bazi podataka MongoDB, kao bazi za pohranu podataka kvalitete rada teleskopa MAGIC. Za iscrtavanje podataka korišten je Dygraph program. Konačno rješenje je trebao izgledati slično kao na slici (Slika 21), gdje je prikazano postojeće grafičko sučelje koje je izrađeno u php programskom jeziku, a kojim se pristupa MySql bazi podataka. Prelaskom na mongo i python okruženje, sustav postaje usklađen sa predloženim tehnologijama koje će se koristiti za CTA. Na ovaj način se u sustavu MAGIC teleskopa može i želi testirati

tehnologije koje se planiraju koristiti za provjeru kvalitete podataka buduće CTA mreže teleskopa.

Novo predloženo rješenje se može vidjeti na poveznici http://symmetry.fesb.hr/MAGICLTM (Slika 22).



Slika 21: Postojeće grafičko sučelje.



Slika 22: Predloženo novo grafičko sučelje.

U web aplikaciji, mongo bazi pristupamo preko PyMonga. Podaci tako zapisani će se spremati u CSV datoteku na serveru. Dygraphs skripta će koristiti te podatke za iscrtavanje grafova. Glavni dijelovi ove aplikacije su template datoteka ovoga projekta i vews.py datoteka.

Template datoteka koju koristimo je mytemplate.pt i sadrži čitav kod aplikacije. U njenom zaglavlju dohvaćamo sve skripte koje koristimo u našemu kodu. Dio koda padajućeg izbornika koji možemo vidjeti na web stranici možemo vidjeti na slici (slika 23). nakon toga na idućoj slici (slika 24) možemo vidjeti Dygraph kod koji služi za iscrtavanje grafova

Slika 23: Kod padajućeg izbornika

Ova funkcija definirana je tako da se aktivira prilikom klika na botun „Run macro“. Unutar koda taj botun definiran je klasom „'.btn.btn-block.pull-left'“. Funkcija kreira novi Dygraph graf koji za prvi argument prima URL CSV datoteke u kojoj se nalaze podaci iz MongoDB baze podataka. Sljedeći argumenti koje Dygraph prima su unutar vitičastih zagrada i određuju funkcionalnosti grafa. Npr. „valueRange“ određuje minimalnu i maksimalnu vrijednost osi ordinate, „title“ prikazuje naslov grafa u vrhu, „dateWindow“ određuje od kojeg do kojeg datuma će biti prikazane vrijednosti na osi apscise.

```
$(function(){
    $('.btn.btn-block.pull-left').click(function(){
        g9 = new Dygraph(
            document.getElementById("graphdiv9"),
            "http://127.0.0.1:6543/csv2",
                {valueRange: [0,50],
                legend:'always',
                title:'Min-Max',
                showRoller:true,
                rollPeriod:14,
                ylabel:'Value',
                 dateWindow:[ Date.parse("2012-12-09"), Date.parse("2013-04-12") ],
                }
        );
    });
});
```

Slika 24: Dygraph

Views.py je glavna datoteka aplikacije i obavlja najveći dio posla aplikacije, poput pristupa bazi, dohvaćanja podataka, renderiranja, spremanja podataka u datoteku itd. Kod integracije okruženju bitno je da se u direktoriju Views.py (Slika 25) temperatur data postavljen na ispravnu putanju i također u liniji response=FileResponse postavimo putanju na

pravi direktorij kako bi se mogao očitati pravi CSV direktorij.

```
import datetime
from pyramid.response import Response
from pyramid.response import FileResponse
from paste.httpserver import serve
import json
import csv
from bson.json_util import dumps
def my_view(request):
    request.db.authenticate('cta_project','fesbstudent11')
    dumps = request.db.mycollection.find_one()
    json_pars = dumps['temperature_details']
    temperature_data = open('C:/Users/Ante/Desktop/teleskop/GamaTeleskop/Website/cta_project/cta_project/CSV files/temperatures.csv', 'w')
    csvwriter = csv.writer(temperature_data)
    count = 0
    for temp in json_pars:
        if count == 0:
            header = temp.keys()
            csvwriter.writerow(header)
            count += 1
        csvwriter.writerow(temp.values())
    temperature_data.close()
    return {'project':'cta_project'}
def csvview(request):
    response = FileResponse(
        'C:/Users/Ante/Desktop/teleskop/GamaTeleskop/Website/cta_project/cta_project/CSV files/temperatures.csv',
        request=request,
        content_type='csv'
        )
    return response
```

Slika 25: Views.py

2.4. Korištenje Githuba za integraciju projekta

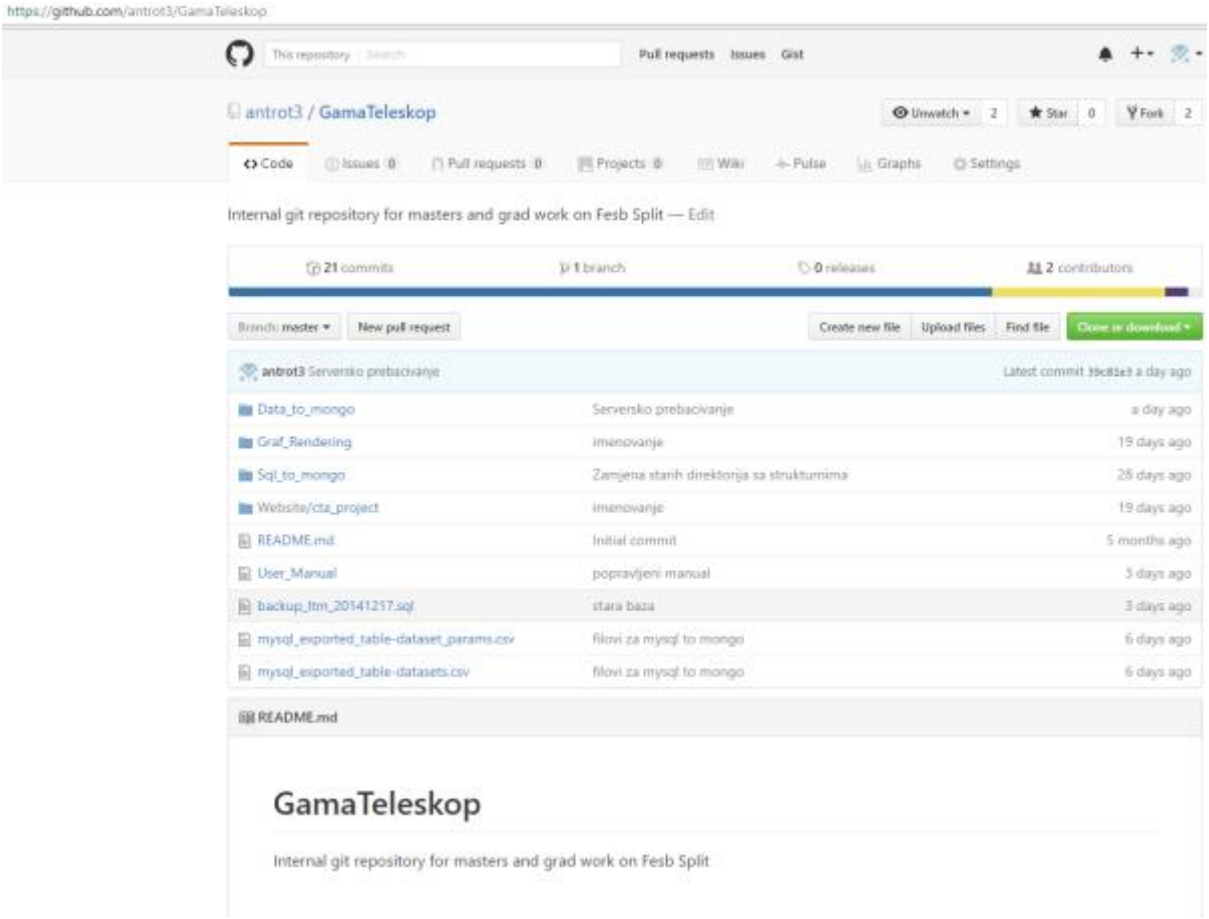
Za strukturiranje i razvoj programskog koda korištena je Git/Github platforma. Cilj je bio da se kod, razvijen od više sudionika, objedini na jednom mjestu, kako bi se mogao strukturirati i lakše dodatno razvijati. Jednako tako, potrebno je imati i mjesto gdje će korisnici moći preuzeti programski paket na korištenje.

Vrlo je česta pojava da se Git i Github smatraju kao jedan program, Github je repozitorij na webu u kojemu su sadržane sve promjene vezane za kod, dok je sam Git zapravo kontrolor verzije sistem (eng. Version controll system), te služi za upravljanje verzijama koda. Ovaj sustav je često prisutan kao okvir za razvoj programskog koda.

Github se u ovom projektu koristi kako bi se na lagan način kontrolirao protok podataka koje stalno izmjenjujemo, također su izrađene i upute za korištenje za potrebe novih korisnika. Repozitoriju se pristupa brzo i pouzdano, a za slučaj da se u lokalnome razvojnome okruženju nešto izgubi ili prebriše, lakoćom se ponovno može postaviti cijeli sustav. Jednako tako se može, ako bude potrebno, vratiti na pojedinu prethodnu verziju koda ikada napisanu. Github repozitorij se lako kreira. Potrebno se registrirati na stranici https://github.com i nakon toga instalirati Git. Nakon instalacije pokrene gitov terminal (git bash), koji možemo pronaći

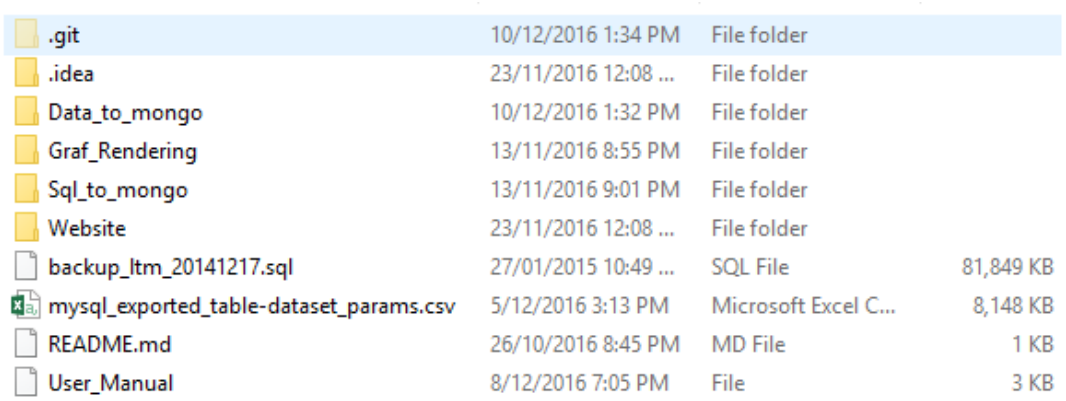
pomoću search polja. U terminalu dođemo do putanje gdje želimo klonirati github repozitorij i samo upišemo naredbu:

`git clone https://github.com/antrot3/GamaTeleskop`.



Slika 26: Github repozitorij.

se može klonirati repozitorij s githuba (Slika 26). Na osobnom/lokalnom računalu repozitorij će izgledati kao na slici (Slika 27).



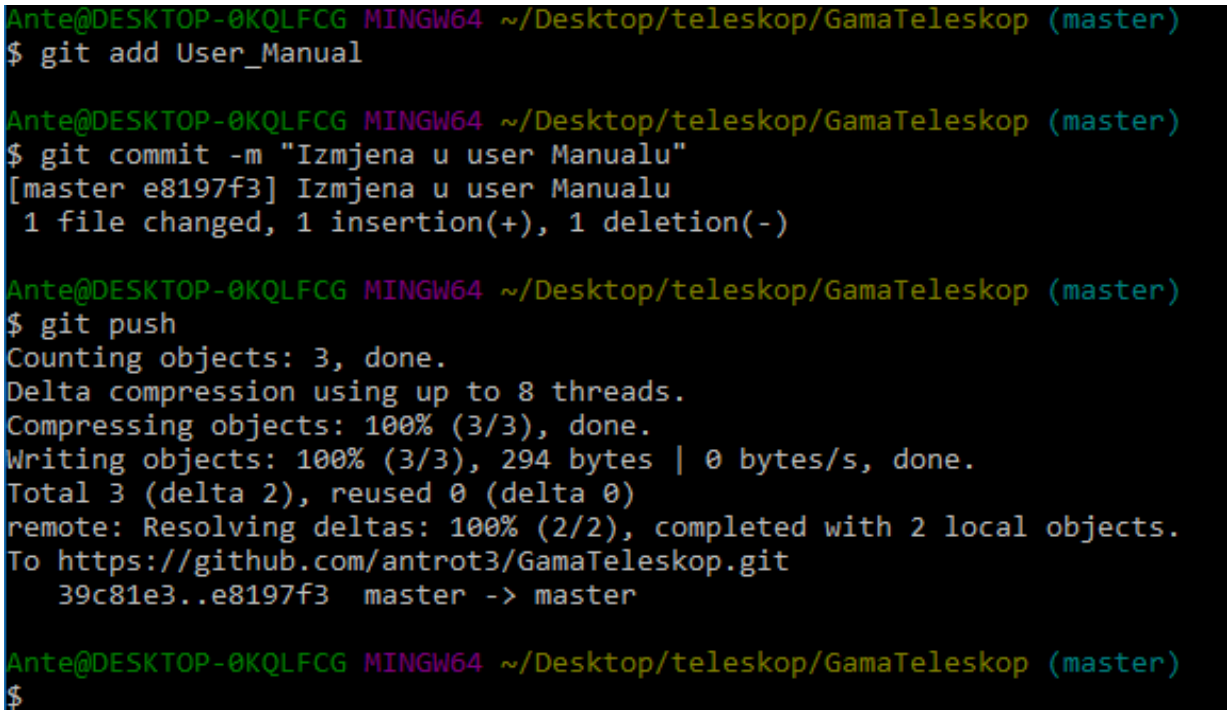
Slika 27: Github repozitorij na lokalnome računalu.

Korištenje cijeloga sustava je jednostavno. Kao što je navedeno, na lokalnom računalu se kreira i preuzme kod s github repozitorij. Nakon što su izvršene i testirane potrebne izmjene i dopune koda na lokalnom računalu, potrebno je promjene unijeti u repozitorij na serveru. Prije unošenja novih detalja, s *git status* naredbom, se može provjeriti usklađenost novog koda s onim na repozitoriju (Slika 28). Unos izmjena se može napraviti s nekoliko jednostavnih naredbi (Slika 29).

```
Ante@DESKTOP-0KQLFCG MINGW64 ~/Desktop/teleskop/GamaTeleskop (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   User_Manual
        modified:   Website/cta_project/.idea/workspace.xml
        deleted:    mysql_exported_table-datasets.csv

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .idea/
        Data_to_mongo/New Text Document.txt
        Website/.idea/
        Website/cta_project/.idea/deployment.xml
        Website/cta_project/.idea/webServers.xml

no changes added to commit (use "git add" and/or "git commit -a")
```

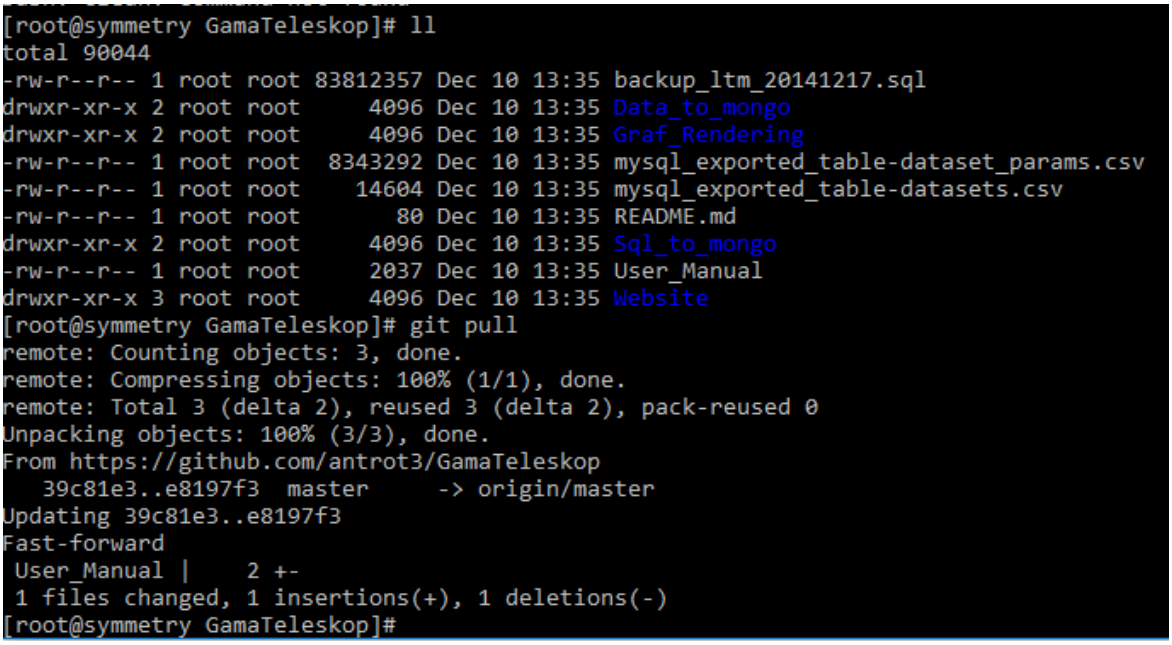
Slika 28: *git status* naredba.

```
git add - za dodavanje svih izmjena ili

git add ime dokumenta – za dodavanje jednog dokumenta

git commit –m ''neka poruka'' – potvrda da se želi spremiti kod

git push – slanje na Github
```



Slika 29: *git add, git commit, git push.*

Nakon unesenih izmjena se s naredbom *git pull* može povući posljednje izmjene, te ih testirati i koristiti (Slika 30).



Slika 30: Git pull na serveru

3. Korištene tehnologije

3.1. HTML

Html (eng. *Hyper Text Markup Language*) [6] je osnovni jezik za izradu web stranica i web aplikacija. Zajedno s CSS-om (eng. *Cascading Style Sheets*) i JavaScriptom čini jedan od tri glavna jezika za izradu web stranica. Web preglednik prima Html dokumente s web servera ili iz lokalnog spremnika i renderira ga u multimedijsku web stranicu. Html opisuje strukturu web stranice.

Html elementi su glavni blokovi za sastavljanje web stranice. Html konstruktima možemo umetati slike i ostale objekte poput interaktivnih formi na web stranicu. To nam uz paragrafe, naslove (eng. *heading*), liste, linkove i ostale objekte dopušta da napravimo strukturirani dokument. Html objekti se označavaju s tagovima. Uzmimo za primjer sliku <img/>. Za razliku od slike, većina drugih objekata ima početni i krajnji tag. Uzmimo za primjer paragraf , gdje prvi tag označava početak, a drugi kraj paragrafa. Web preglednik ne prikazuje tagove ali ih koristi za interpretaciju stranice.

Prvi dostupan opis HTML-a je dokument „HTML tags“, a sam naziv prvi put spominje Tim Berners-Lee krajem 1991. Prva verzija Html-a je objavljena 1993. godine. U tu osnovnu verziju se nije moglo dodati slike. Prva standardna verzija Html-a je bila inačica 3.0 koja je imala mogućnost dodavanja tablica. No, sadašnja verzija Html-a, verzija 5, donosi brojne nove mogućnosti koje HTML 4.01 i XHTML 1.x nisu imali, kao što je mogućnost reprodukcije videozapisa na stranicama bez korištenja Adobe Flasha ili Microsoft Silverlighta, mogućnost upravljanja pomoću tipkovnice i opcije za bilo koju vrstu manipulacija, drag and drop, canvas kao i ostali novi elementi. Kada pišemo novi Html dokument obično koristimo tekstualne editore kao što su Notepad ili Notepad++ ili neki drugi. Postoje također CMS (Content Management System) sustavi koji već u sebi imaju ugrađene funkcionalnosti Html-a, pa tako uvođenjem tih sustava sada svatko može, relativno lako, napraviti svoju web stranicu.

3.2. CSS (Cascading style Sheets)

CSS je [6] jezik za oblikovanje i opisivanje dokumenata koji su napisani u markup jezicima kao što je HTML. Iako se najviše koristi za izradu vizualnog stila web stranice i korisničkih sučelja u HTML-u, ovaj jezik se može također koristiti XML dokumente (eng. *Extensible Markup Language*). Uz HTML i JavaScript, CSS je ključan jezik za izradu vizualno atraktivnih web stranica, korisničkog sučelja za web aplikacije i korisničkog sučelja za mnoge mobilne aplikacije.

Primarna svrha za koju je napravljen CSS je da se odvoji sadržaj dokumenta od načina njegovog prezentiranja, uključujući raspored elemenata, boju i fontove. Razdvojenost ova dva bloka daje pristupačniji sadržaj, više kontrole i fleksibilnosti nad prezentacijom i karakteristikama web stranice, što omogućava višestrukim HTML stranicama da dijele isti format koji je deklariran unutar CSS dokumenta. Na ovaj način nije potrebno stalno ponavljati isti kod na više različitih mjesta nego je sve posloženo na jednom mjestu. Ovo je također korisno jer se može definirati responzivnost stranice. Responzivnost stranice je mogućnost da se stranica prikaže drugačije od originalne ovisno o veličini zaslona web preglednika, što je postalo jedna od ključnih stvari u razvoju web stranica i aplikacija otkad su na tržište izašli pametni telefoni i tableti čija je veličina zaslona znatno manja od zaslona klasičnog stolnog računala.

Lista selektora u CSS-u:

- Selektor elementa (p{} u ovome slučaju elementi p odnosno svi paragrafi koji nisu drugačije označeni), ovaj selektor ima najniži prioritet i izvršit će se za taj element jedino u slučaju kada nisu ostali selektori specificirani
- Selektor klase (.para2{} u ovom slučaju u Html-u se označi o kojoj se klasi radi class="para2" i poziva se unutar CSS-a koji se piše unutar <style> elementa, uz pomoć .para2). Ovaj selektor ima srednji prioritet.
- Selektor id (#para1{} također se radi o p elementu ali ovaj put uz pomoć id="para1" u ovome slučaju označi o čemu se radi). Ovaj selektor ima najviši prioritet važnosti.

Navedeni slučaj je jednostavan i koristi se za promjenu boja i poravnanje po centru slova, ali CSS je puno moćniji od toga i može se koristiti za oblikovanje gotovo svih detalja i na razne načine na web stranicama i web aplikacijama.

3.3. JavaScript

JavaScript [6] je dinamički jezik koji se koristi u većini današnjih web stranica. Svi poznati preglednici ga podržavaju i jako je prikladan kako bi se stranica učinila što dinamičnijom. Ovaj programski jezik se izvršava u web pregledniku na strani korisnika. Napravljen je da bude sličan Javi radi lakšeg korištenja, ali nije objektno-orijentiran kao Java, već se temelji na prototipnom nasljeđivanju i tu prestaje svaka veza s programskim jezikom Java. Izvorno ga je razvila tvrtka Netscape. Javascript je primjena ECMAScript standarda. Javascript zajedno s AJAX (eng. *Asynchronous JavaScript and XML*) tehnikom čini web aplikaciju interaktivnijom i lakšom za korištenje.

JavaScript većinu svoje sintakse bazira na programskom jeziku C. Koristan je za dinamičko mijenjanje sadržaja na stranici, primjerice, klikom na neki HTML element. U našem slučaju, koristimo ga za dohvaćanje raznih informacija iz baze podataka, koje nakon toga pomoću alata za iscrtavanje grafova prikazujemo na web stranici.

3.4. Python

Python [7] je objektno-orijentiran programski jezik koji se može usporediti s Perlom, Rubyem, Schemom ili Javom po automatskoj memorijskoj alokaciji. Stvori ga je Guido van Rossum 1990. godine (prva bolja verzija je objavljena 1991. godine), a ime je dobio po televizijskoj seriji Leteći cirkus Montya Pythona. Python dopušta programerima nekoliko stilova programiranja. Objektno orijentirano, strukturno i aspektno orijentirano programiranje je moguće ostvariti korištenjem Pythona i ova fleksibilnost ga čini sve popularnijim programskim jezikom. Python se većinom koristi na Linuxu, ali postoje i verzije za ostale operacijske sustave poput MacOS i Windowsa.

Koristi elegantnu sintaksu što čini program koji pišemo čitljivijim. Jednostavan je jezik koji pojednostavljuje put do efikasnoga programa. Ovo čini Python idealnim za razvoj prototipova i ostalih zadataka koji se moraju brzo izvršiti, bez dovođenja u pitanje stabilnosti koda. Dolazi pred instaliran sa mnogim jezičnim bibliotekama koje podržavaju uobičajene zadatke poput spajanja na web server, pretraživanje teksta s regularnim izrazima, čitanje i modificiranje datoteka. Python interaktivni mod daje nam jednostavni način za testiranje

kratkih dijelova programa. Postoji također okruženje za paketni razvoj (eng. *bundled development*) koje se zove IDLE. Lako ga je prilagodit pomoću modula u programske jezike kao što su C ili C++. Također se može ugraditi u aplikaciju kako bi dobili programabilno sučelje. Besplatan je.

Unutar programerske zajednice česte su kritike Pythona na račun njegove izvedbe. Pošto je Python interpreterski jezik, programi koji su napisani u njemu vrte se nešto sporije nego oni napisani u kompajlerskim jezicima, kao što su C, C++ itd. Međutim, unatoč brzinskoj manjkavosti u programiranju se Python znatno koristi. Python se često uspoređuje s Javom - oboje su interpreterski jezici i oba programa nemaju gotovo nikakvu podršku za višejezgreno izvođenje programa, pošto i Python i Java koriste samo jednu procesorsku jezgru. Java je jezik koji se dosta primjenjuje u izradi interaktivnog web sadržaja i mobilnih aplikacija, dok je Python gospodar PC programa. Kada je u pitanju brzina, Python i Java su prilično jednaki.

Neke od ključnih riječi u Pythonu :

- If izraz, drži kod pod nekim uvjetom
- For izraz, koristi standardnu for petlju
- While, također standardna petlja u programiranju koja se vrti određen broj puta, sve dok je tvrdnja zadana u njoj istinita
- Try
- Class, izraz za definiranje klasa
- Def izraz, koji definira funkciju ili metodu
- Assert, koji se koristi u debugiranju kako bi se provjerili određeni uvjeti
- Import, koji se koristi pri uključivanju dodatnih vanjskih modula

3.5. Pyramid

Pyramid [8] je prvotno potekao iz Pylons projekta. Pyramid je web okruženje otvorenog koda, napisano u Pythonu i bazirano je na WSGI tehnologiji. WSGI (eng. *Web Server Gateway Interface*) je aplikacija za jednostavano i univerzalno korisničko sučelje između web servera i web aplikacije ili okvira za Python programski jezik. Od 2003. godine je postao standard za razvoj Python web aplikacija. WSGI ima dvije strane - serversku stranu (obično je to Apache server) i aplikacijsku koja se sastoji od Python skripti. Procesiranjem WSGI zahtjeva serverska

strana izvršava aplikaciju i daje podatke o okruženju, te poziva callback funkciju na strani aplikacije. Aplikacija procesira zahtjev i vraća odgovor na serversku stranu koristeći callback funkciju koja joj je dana. Srednji sloj (eng. *middleware*) se nalazi između aplikacije i servera i odgovoran je za komunikaciju između njih.

Pyramid je minimalistički, objektno orijentiran MVC razvojni okvir. Može se integrirati zajedno s SQL bazom, također se može koristiti i druge sustave za upravljanje bazama podataka, kao što su Zope Object, razne NoSql baze, CouchDB ili u našem slučaju MongoDB.

Python nam također dopušta da definiramo rute za regularne izraze koji mapiraju objekte, kao i većina drugih razvojnih okruženja.

3.6. Sql baza podataka

Sql je jedan od prvih jezika za relacijske baze podataka. Njegov model je predstavio Edgar F. Codda u svome radu „A Relational Model of Data for Large Shared Data Banks“, koji je izdan 1970. u časopisu Association for Computer machinery (ACM), Comunications of the ACM. Sql je postao jedan od najčešće korištenih jezika za relacijske baze podataka, te se može reći da je Coddov model široko prihvaćen kao definitivni model za relacijske baze podataka. Na slici (Slika 31) možemo vidjeti model baze.

#	Name	Type	Collation	Attributes	Null	Default
1	ID	bigint(20)		UNSIGNED	No	None
2	user_login	varchar(60)	utf8mb4_unicode_ci		No	
3	user_pass	varchar(255)	utf8mb4_unicode_ci		No	
4	user_nicename	varchar(50)	utf8mb4_unicode_ci		No	
5	user_email	varchar(100)	utf8mb4_unicode_ci		No	
6	user_url	varchar(100)	utf8mb4_unicode_ci		No	
7	user_registered	datetime			No	0000-00-00 00:00:00
8	user_activation_key	varchar(255)	utf8mb4_unicode_ci		No	
9	user_status	int(11)			No	0
10	display_name	varchar(250)	utf8mb4_unicode_ci		No	
11	Company	tinyint(2)			Yes	3

Slika 31: Model baze podataka u Sql-u

Sql upiti (eng. *queries*) su najčešće operacije u Sql-u. Upit koristi SELECT naredbu, koja vraća podatke iz jedne ili više tablica. Standardna SELECT naredba nema nikakav utjecaj na bazu

podataka (neće je izmijeniti). No, neke nestandardne SELECT naredbe kao što je SELECT-INTO kod nekih baza podataka mogu mijenjati i podatke u njoj. Upiti omogućuju korisnicima da odaberu željene podatke. Upit nam daje listu stupaca koji će biti dio krajnjeg rezultata upita. Za odabiranje svih stupaca u tablici koristimo ('*') operator. SELECT je najsloženiji upit u Sql i ima svoje opcionalne podfunkcije kao što su:

- FROM izraz, koji govori iz koje će se tablice prihvatiti podatak
- WHERE izraz, koji daje mogućnost da usporedimo neki izraz i prema tome dobijemo rezultate. Primjerice, imamo tablicu studenata i ocjena nazvanu Studenti. Kako bismo ispisali sve studente s ocjenama poviše 3, upisali bi SELECT * from Studenti where ocjena>3
- GROUP BY govori kako ćemo grupirati određene podatke
- ORDER BY izabiremo stupac po kojem će se poredati podatke
- DISTINCT eliminira duplicirane podatke, odnosno vraća samo jedinstvene retke

Manipuliranje podacima je još jedna jako važna stvar ako se misli koristiti bazama podataka. DML (eng. *Data Manipulation Language*) je dio Sql jezika koji služi za dodavanje, mijenjanje i brisanje podataka unutar baze. Insert operacija dodaje podatke u tablicu. Treba se pripaziti da se tekstualne vrijednost (eng. *string*) pišu unutar jednostrukih navodnika. Update naredba modificira neki od već postavljenih redaka. Merge operacija se koristi da se kombiniraju podaci iz više tablica. Na kraju još ćemo spomenuti DDL (eng. *Data Definition Language*) naredbe koje se koriste za definiranje modela tablice (Slika 34). Osnovne naredbe su CREATE naredba koja radi novu tablicu i preko nje se definira struktura i model nove tablice. ALTER mijenja tablicu tako da se doda novo polje ili makne neko od starih, RENAME naredba mijenja ime nekog od stupaca. DROP briše tablicu.

3.7. MongoDB baza podataka

MongoDb [9] je u jako malo vremena postala jako popularna baza podataka za web aplikacije i savršeno se uklapa u Node.JS aplikacije, te dopušta pisanje JavaScript programa na strani klijenta, bekendu i sloju baze podataka. Njena nezahtjevna priroda je savršena za podatke koji se stalno mijenjaju i razvijaju, te je zavidna lakoća kojom se može mijenjati struktura tablica. Ovo je za razliku od Sql (kod koje treba zahtjevno planiranje prije izvedbe) jednostavna i brza platforma u kojoj se s lakoćom može promijenit model tablice i razno razne stvari u

budućim verzijama baze podataka. U početku je malo problematično za postaviti ovu bazu podataka. Ovaj projekt je razvijan djelomično na Linuxu i djelomično na windowsima 10, ali je instalacija i postavljanje na oba OS dosta slična.

U mongu možemo pisati funkcije i, kao što je navedeno, kompatibilan je i sličan JavaScriptu.

Funkcija:

```
Function times2(broj){

Return num*2;

}

Times2(5) // vraća 10
```

Korisne naredbe:

- Show dbs- popis svih baza u mongu
- db- baza koju trenutno koristimo
- use test12- stvara i koristi novu bazu pod imenom test12
- db.test12.insert(mate)- dodaju u bazu test12 dokument pod imenom mate
- db.test12.find()- izlistava sve podatke iz baze
- db.test12.find().pretty()- daje jasniji/ljepši pregled baze
- db.test12.update({'name':'ante'}, {'name':'jure'}) služi za dodavanje novih podataka u već postojeću bazu ili prepravak starih podataka, dakle u našem slučaju je u bazi test12 potražen ante i njegovo ime je promijenjeno u jure.
- db.test12.remove- radi na istom principu kao i update samo briše podatke
- dbquery.shellBatchSize =100000 – povećava broj išpartanih redova po upitu

Tipovi podataka:

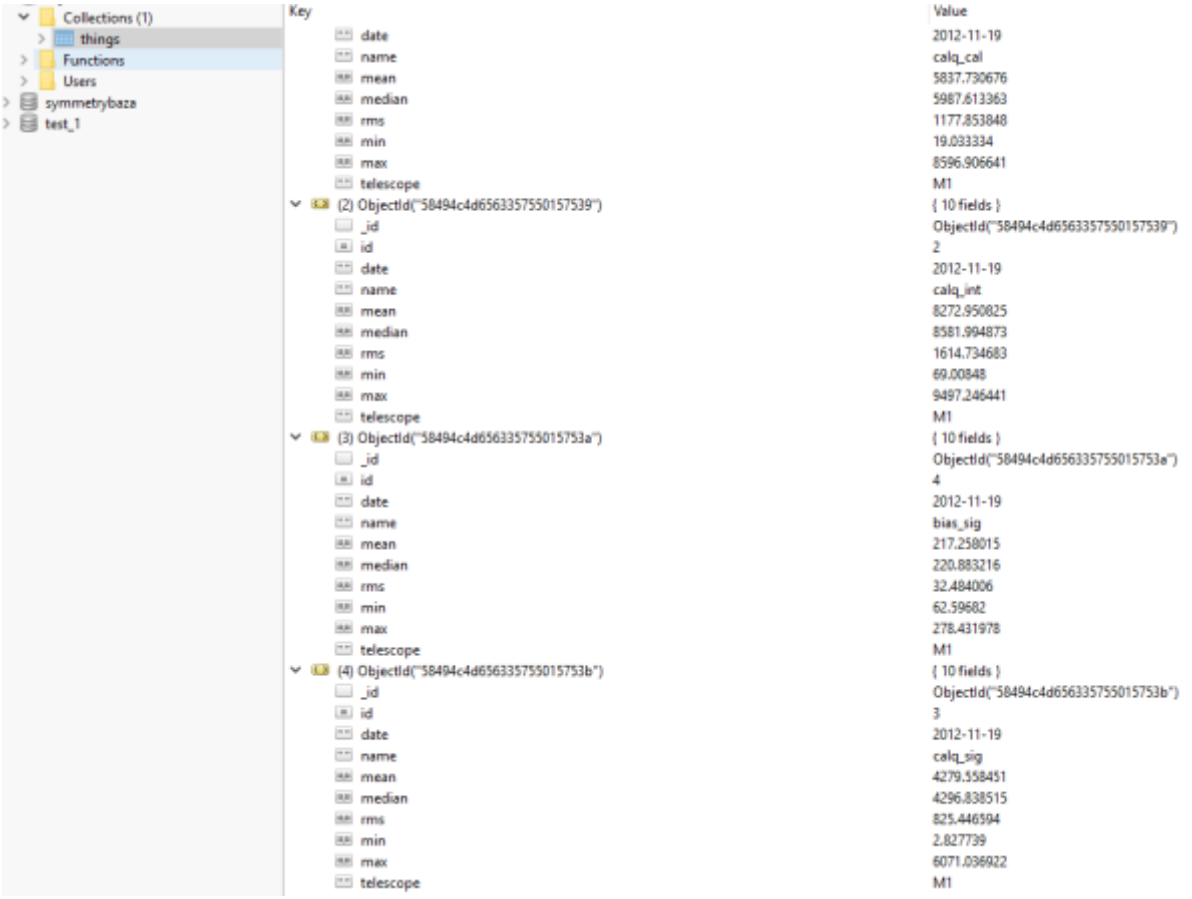
Ovdje je važno imati na umu kada radimo s MongoDB, da postoji mala razlika kod mapiranja podataka između onih koje podupire MongoDb (slika 32) i onih ugrađenih u JavaScript. Ovo su neki od njih:

- **Float** je 8-bitni tip koji se direktno pretvara u JavaScript broj

- **Double class** je posebna klasa koja predstavlja float vrijednost, ovo je posebno korisno kada koristimo kolekcije koje su prilično popunjene i gdje moramo osigurati da su uvijek float vrijednosti
- **Integers** je malo zahtjevniji zbog činjenice da JavaScript predstavlja sve brojeve u 64 bitnom float formatu, što znači da je maximum integer vrijednosti 53 bita. Mongo ima dva tipa integera, 32 bitni i 64 bitni. Mongo prvo pogleda da li može spremiti broj u 32 bitnu verziju, a ako ne može onda ga pretvara u 64 bitnu verziju. Isto proba sa 53 bitnom vrijednošću, ako ne uspije vraća LONG(64 bitnu verziju) kako ne bi izgubio preciznost.
- **Long class** je posebna klasa koja dopušta spremanje 64 bitnog integera
- **Date** mapira direktno Javascript datum.
- **RegExp** mapira direktno u Javascrip RegExp
- **String** mapira direktno u JavaScript string(kodiran u UTF8)
- **Binarry class** posebna klasa koja dopušta spremanje podataka u MongoDb
- **Code class** posebna klasa koja dopušta spremanje JavaScript funkcija u MongoDB, također može dati mogućnosti scopa da se pokrenu metode
- **ObjectID class** posebna je klasa koja sadrži ID mongo dokumenta (ekvivalent primarnome ključu)
- **DefRef class** posebna je klasa koja dopušta uključivanje referenci koje pokazuju na drugi objekt
- **Symbol Class** je specijalna klasa koja dopušta specificiranje simbola, ovo nije bitno za JavaScript, ali jest važno za jezik koji podupire koncept simbola

Postoje i neke funkcije koje modificiraju samo dijelove dokumenata.

- **$inc-** inkremetira određenu vrijednost za određeni broj koji mi zadamo
- **$set-** postavlja varijablu koju zadamo
- **$unset-** briše određeno polje
- **$push-** dodaje vrijednost na kraj dokumenta
- **$pushall-** dodaje nekoliko vrijednosti na kraj dokumenta
- **$addToSet-** dodaje vrijednost u neki niz jedino ako ta vrijednost već ne postoji u nizu
- **$pop-** briše zadnji element u nizu
- **$pull-** briše vrijednosti iz postojećeg niza
- **$pullAll-** briše nekoliko vrijednosti iz postojećeg niza
- **$rename-** mijenja ime polja
- **$bit-** radi bitovne operacije (1 ili 0, true ili false)



Slika 32: Pregled polja i njihovih vrijednosti unutar mongo baze.

3.8. ROOT

Root [10] je okruženje za procesiranje podataka, kreirano na CERN-u, u središtu istraživanja visoko energetske fizike. Svaki dan tisuće fizičara koriste ROOT aplikaciju za analiziranje podataka ili za izvedbu simulacija. Sa ROOT-om se može:

- **Spremati podatke.** Mogu se spremati podaci u komprimiranoj binarnoj formi u ROOT dokumentu. Format objekta je također spremljen u istom dokumentu. Čak i u slučaju kada izvorni dokumenti koji opisuju model podataka nisu dostupni, informacije zapisane u ROOT dokumentima će se moći pročitati. ROOT pruža strukturu podataka, stablo, što je jako moćan alat za brz pristup golemoj količini podataka, brže nego pristupanje normalnom dokumentu.
- **Pristup podacima.** Podacima zapisanim u jedan ili više ROOT dokumenata možemo pristupiti s osobnog računala, sa web-a ili velikog sistema koji raspoređuje dokumente, na primjer GRID. ROOT stabla se mogu koristiti u nekoliko dokumenata, te se mogu

spojiti i pristupati jedinstvenim objektima. To omogućuje pregledavanje ogromne količine podataka.

- **Kopanje podataka(eng. Data Mining).** Moćni matematički i statistički alati su dostupni kako bi se njima manipuliralo podacima. Puna snaga C++ aplikacije i paralelnog procesiranja je dostupna za bilo koju vrstu manipuliranja podacima. Podaci također mogu biti generirani sljedeći statističku distribuciju i model, što omogućuje simuliranje kompleksnih sistema.
- **Objavljivanje rezultata.** Rezultati se mogu prikazati histagramom, raznim krivuljama, funkcijama. ROOT grafikoni se mogu prilagoditi u realnom vremenu u samo nekoliko klikova mišem. Oblici dovoljno kvalitetni za publiciranje mogu se spremiti u PDF ili nekome drugome postojećem formatu.
- **Interaktivno pokretanje i izrada vlastitih aplikacija.** Može se koristiti Cling C++ intepreter za interaktivne programe i za pisanje macroa, ili se može kompajlirati program radi bržeg izvođenja.
- **Koristi ROOT zajedno s ostalim jezicima.** Root pruža alate kako bi se jednostavno integrirao sa već postojećim programskim jezicima kao što su Python, Ruby i Matlab.

Instalacija samoga ROOT-a je dosta zahtjevna, ali su dostupne detaljne upute za različite verzije operativnih sustava.

3.9. GIT

Git [11] je sustav za upravljanje izvornim kodom (eng. Version control system) koji se koristi za razvoj raznih programa i aplikacija. Njegove prednosti su brzina, očuvanje izvornog koda i nelinearni rad. Git je napravio Linus Torvalds 2005. godine za razvoj Linux karnela, gdje su također programeri na karnelu pripomogli razvoju git-a. Kao i većina version control systema, ali za razliku od client-server systema, svaki direktorij na svakom računalu sadrži cjelovitu povijest i sve prijašnje verzije koda, neovisno o pristupu internetu ili centralnom serveru. Karakteristike:

- Podupire ne linearni razvoj i podijeljeni razvoj: Git podržava brzo razdvajanje na grane (uzmemo na primjer dva programera rade svaki na svome dijelu koda i ne želimo da se jedan drugome stvaraju probleme tako da zadiru u tuđi kod, tada napravimo dvije grane i podijelimo ih svakom programeru i na kraju kada su oba djela

koda gotova spojimo ih u glavnu granu) i njihovo spajane, svaka promjena koju programer napravi na svojoj grani ne mijenja čitav kod nego samo onaj direktorij u kojemu se izvršila promjena. Također ako se desi da su se ista datoteka promijenila, tada moramo rješavati konflikte(trebamo izabrati koje dijelove koda ćemo koristit u krajnjem kodu) koji se dešavaju prilikom spajanja grana. Za to se koriste conflict resolution addoni.

- Iznimno koristan za velike projekte: Bilo koji projekt se može podijeliti na onoliko dijelova koliko i programera imamo, ako je to potrebno

Osnovne naredbe:

- git clone (putanja do repozitorija)
- git config --global user.name "(username)"
- git config --global user.email "(to je email adresa)" – ove dvije naredbe će se vjerojatno zatražiti prilikom kloniranja nekoga repozitorija.
- git add (ime direktorija) ili git add . za dodavanje svega na stage
- git commit –m "(poruka)“ – potvrđuju se promjene ali se još ne dodaju na repozitorij
- git push origin master – šalje promijene na master branch
- git status – daje nam popis dokumenata koji su izmijenjeni i ili koji se moraju dodati sa git commitom ili git add-om
- git pull – dohvaća promijene na serveru u radni direktorij

Github [12] je web repozitorij koji služi za spremanje koda. Možemo napraviti privatni repozitorij koji se plaća ili otvoreni repozitorij. Većina open source programa je pospremljena na githubu i besplatna za preuzeti.

4. Zaključak

Zadatak ovog diplomskog rada je bio konkretno doprinijeti razvoju programske podrške sustava dugotrajne (LongTerm), višednevne, provjere kvalitete podataka MAGIC/CTA (Cherenkov Telecope Array) teleskopa. Sustav provjere kvalitete podataka je vrlo važan dio svakodnevnog rada MAGIC teleskopa i buduće mreže gama teleskopa CTA. Povratna informacija o mogućim problemima je ključna u otklanjanju poteškoća u radu ovih teleskopa koji mjere najenergetskiji dio elektromagnetskog spektra - gama područje.

Ovaj rad je dio timskog projekta koji je za cilj imao izmijeniti postojeću programsku tehnologiju dugotrajnog praćenja kvalitete podataka MAGIC teleskopa sa tehnologijama koje se planiraju koristiti za CTA mrežu. Prvi konkretan zadatak ovog tima je bio prebaciti postojeću MySql bazu podataka na MongoDB sustav. Drugi je zadatak bio omogućiti spremanje podataka koje MAGIC dnevno proizvodi za dugotrajnu kontrolu kvalitete podataka u mongo format. Treći zadatak je bio izrada web sučelja u kojem će se, korištenjem alata za iscrtavanje, prikazivati navedene podatke. Na kraju, potrebno je bilo integrirati pojedine dijelove u operativnu cjelinu. Moj doprinos je bio u svezi prva dva zadatka, te integracija sustava.

Kako je navedeno u radu, dio zadatka prebacivanja stare Sql baze u Mongo je uspješno odrađen u koristeći Mongify i PhpMyAdmin alate. Jednako tako, koristeći elegantan JavaScript program uspješno su svakodnevni podaci za dugotrajno praćenje kvalitete rada MAGIC teleskopa iz formata tekstualne datoteke zapisani u MongoDB bazu podataka. Svi gore navedeni koraci su integrirani na linux server, sličan onom koji se koristi na La Palmi za MAGIC. U sklopu ovog projekta, kao dio tima sam se upoznao također s izradom web sučelja za prikaz podataka i iscrtavanje grafova iz dobivenih parametara.

Idući koraci bi bili vezani uz daljnji razvoj sustava. Planira se rad na razvoju web sučelja, testiranju i implementaciji dodatnih paketa za iscrtavanje (kao amCharts), te implementacija postignutih rješenja u službeni MAGIC sustav za dugotrajnu provjeru kvalitete podataka.

5. Literatura

[1] "How CTA will detect Cherenkov light", s Interneta,

https://www.cta-observatory.org/about/how-cta-works/ , 10.10.2016

[2] "CTA", s Interneta, https://www.cta-observatory.org/, 10.10.2016

[3] "MAGIC projekt", s Interneta, https://symmetry.fesb.hr/magic/, 10.10.2016

[4] Mazin, D: " Upgrade of the MAGIC telescopes"

[5] " Mongify", s Interneta, http://mongify.com/, 10.10.2016

[6] "HTML", "CSS", "JavaScript", s Interneta,

http://www.w3schools.com/ , 10.10.2016

[7] "Python", s Interneta, https://www.python.org/ , 10.10.2016

[8] "The Pyramid Web Framework", s Interneta,

https://pyramid.readthedocs.io/en/latest/, 10.10.2016

[9] " MongoDB", s Interneta, https://www.mongodb.com/, 10.10.2016

[10] " About ROOT", s Interneta, https://root.cern.ch/, 10.10.2016

[11] "Git", s Interneta, https://en.wikipedia.org/wiki/Git, 10.10.2016

[12] "Github", s Interneta, https://en.wikipedia.org/wiki/GitHub, 10.10.2016

6. Popis Kratica

- CSS – Cascading Style Sheets
- CTA – Cherenkov Telescope Array
- HTML – HyperText Markup Language
- MAGIC – Major Atmospheric Gamma Imaging Cherenkov Telescopes
- PHP – Hypertext Preprocessor
- SQL – Structured Query Language
- WSGI – Web Server Gateway Interface