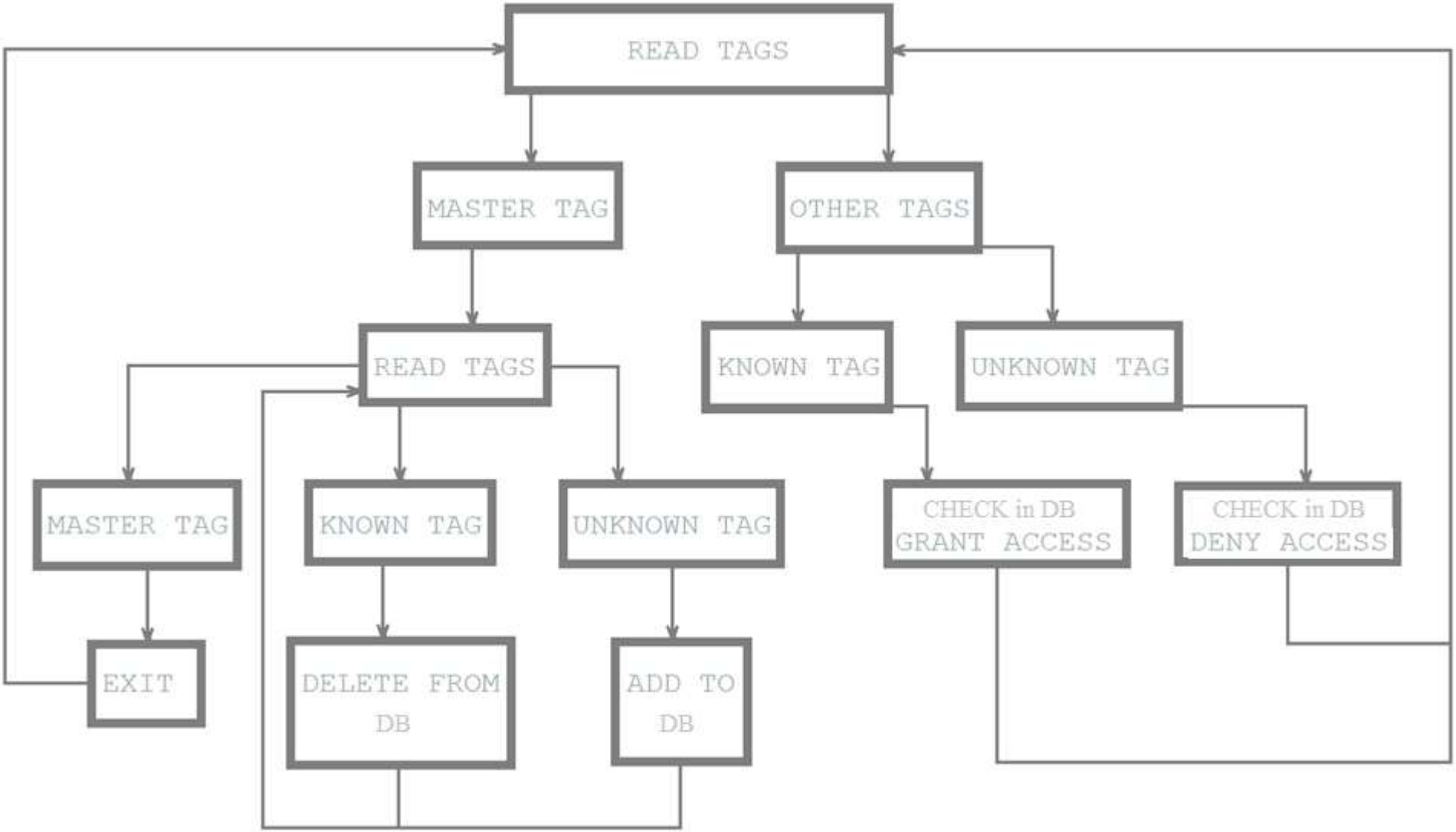
IDENTIFIKACIONI SISTEMI S BAZOM PODATAKA

ID UREĐAJ - DIJAGRAM TOKA



POVEZIVANJE ID UREĐAJA I BAZE PODATAKA

HTTP POST

1 /resource

Body URL Encoded
api_key=**API_KEY**&field1=**30**

HTTP Response

2 Status 200 (OK)

HTTP – za komuniciranje između servera i klijenta

- HTTP – Hypertext Transfer protokol
- Dizajniran da omogući komunikacju između servera i klijenta
- Protokol zahtjeva i odgovora
- Klijent šalje HTTP zahjev serveru – server klijentu uzvraća odgovor
- Odgovor sadrži status izvršenja zahtjeva, a može sadržati i dodatne podatke.

- U radu sa ThingSpeak platformom Arduino uređaj će imati ulogu klijenta a ThingSpeak platforma ulogu servera.

HTTP zahtjev

HTTP zahtjev generiše klijent, prema imanovanom host-u, lociranom na serveru.

Cilj zahtjeva je pristup resursu na serveru.

Korektno sastavljen HTTP zahtjev sadrži sljedeće elemente:

- Liniju zahtjeva;
- HTTP zaglavlja;
- Tijelo poruke, ako je potrebno.

Nakon svakog HTTP zaglavlja slijedi znak za povratak na početak reda (carriage return) i znak za prelazak u novi red (line feed) (CR-LF). Nakon poslednjeg zaglavlja dodatni CR-LF je dodat (za dobijanje prazne linije), nakon kojeg počinje tijelo poruke.

HTTP zahtjev – Linija zahtjeva

Linija zaglavlja je prva linija u poruci zahtjeva. Sastoji se iz tri dijela:

- Metod. Metod je jedno-rječna komanda koja govori serveru što da radi sa resursom. Na primjer, server može biti upitan da pošelje resurs klijentu.
- Komponenta staze URL-a za zahtjev. Staza identifikuje resurs na serveru.
- Broj HTTP verzije, ukazuje na HTTP specifikaciju s kojom je klijent pokušao uskladiti poruku.

Primjer linije zahtjeva:

GET /software/htp/cic/indeks.html HTTP/1.1

Linja zahtjeva može sadržati i dodatne podatke.

HTTP zahtjev – Zaglavlje (Header)

- Pruža prijemnoj strani informacije o poruci, pošiljaocu i načinu na koji pošiljaoc želi da komunicira sa primaocem.
- Svako HTTP zaglavlje se sastoji od imena i vrijednosti.
- HTTP protokol definiše standarsni set HTTP zaglavlja i opisuje kako ih koristiti korektno.
- HTTP zaglavlje zahtjeva klijenta sadrži informacije koje server može upotrijebiti u odlučivanju kako da odgovori na zahjev. To može biti da klijent čita zahtijevani dokunet na francuskom ili njemačkom jeziku i da dokument treba biti poslat jedino ako je mijenjan od naznačenog datuma.

```
Accept-Language: fr, de
If-Modified-Since: Fri, 10 Dec 2004 11:22:13 GMT
```

HTTP zahtjev – Tijelo poruke

- Može se nazvati i tijelom zahtjeva
- Aktuelni sadržaj poruke.
- Tijelo poruke može biti u originalnom obliku ili može biti kodirano.
- Može se nazvati i tijelom zahtjeva
- Prikladno je za neke metode zahtjeva, dok za druge nije.
- Na primjer, zahtjev sa POST metodom, koji šalje ulazne podatke serveru, ima tijelo poruke, koje sadrži te podatke.
- Zahtjev sa GET metodom, koji od servera traži da pošalje resurs, ne sadrži tijelo poruke.

HTTP odgovor

- HTTP odgovor generiše server i šalje klijentu.
- Cilj odgovora je da obezbijedi klijentu treženi resurs ili da ga informiše o izvršenju zahtejva ili da dojavi da je došlo do greške.
- HTTP odgovor se sastoji iz:
  - Statusne linije;
  - Zaglavlja;
  - Tijela poruke, koje je obično neophodno.

Nakon svakog HTTP zaglavlja slijedi znak za povratak na početak reda (carriage return) i znak za prelazak u novi red (line feed) (CR-LF). Nakon poslednjeg zaglavlja dodatni CR-LF je dodat (za dobijanje prazne linije), nakon kojeg počinje tijelo poruke.

HTTP odgovor - Statusna linija

- Statusna linija je prva linija u odgovoru. Sasloji se iz tri segmenta:
  - Broj HTTP verzije, koji ukazuje na HTTP specifikaciju po kojoj je server pokušao da usladi odgovor.
  - Statusni kod, koji je trocifarski broj i ukazuje na rezultat izvršenja zahtjeva.
  - Fraza razloga, poznata i kao tekst statusa, koji je čitljiv čovjeku i sažima značenje statusnog koda.

Primjer statusne linje:

```
HTTP/1.1 200 OK
```

HTTP odgovor – Zaglavlja (Headers)

- Sadrži informacije koje klijent koristi da pronađe više podataka o odgovoru, kao i da pronađe podatke o serveru koji je poslao poruku.
- Ove informacije mogu pomoći klijentu u prezentaciji odgovora korisniku.
- Na primjer, prikazana zaglavlja govore klijentu kada je odgovor poslat, od strane kojeg servera je poslat, kao i da je to JPEG slika.

```
Date: Thu, 09 Dec 2004 12:07:48 GMT
Server: IBM_CICS_Transaction_Server/3.1.0(zOS)
Content-type: image/jpg
```

HTTP odgovor – Tijelo poruke

- Naziva se i tijelom odgovora.
- Većina odgovora sadrže tijelo poruke. Izuzeci su kada server odgovara na zahtjev klijenta, koji je koristio HEAD metod (koji koristi zaglavlja ali ne i tijelo odgovora) i gdje server koristi određene statusne kodove.
- U odgovoru na uspješno izvršen zahtjev, tijelo poruke sadrži resurs koji je klijent zahtijevao ili neke informacije o statusu radnje koju je klijet zahtijevao.
- U odgovoru na neuspješno izvršen zahtjev, tijelo poruke može da pruži dodatne informacije o razlozima greške ili o nekoj radnji koju klijent treba da preduzma da bi se zahtjev uspješno izvršio.

HTTP zahtjev - metode

- HTTP definiše set metoda (načina) da indicira akciju koja će biti izvršena na datom resursu.
- Mada mogu biti i imenice, metode zahtjeva se često označavaju kao HTTP glagoli.
- Svaki metod koristi različitu semantiku.

HTTP zahtjev – vrste metoda

- **GET** Get metod često zahtijeva reprezentaciju navedenog resursa i samo vraća podatke.
- **HEAD** HEAD metod očekuje odgovor identičan GET zahtjevu, ali bez tijela odgovora.
- **POST** POST metod šalje entited specificiranom resursu, često izazivajući promjene stanja ili druge efekte na serveru.
- **PUT** PUT metod sve tekuće prikaze ciljnog resursa sa sadržajem zahjeva.
- **DELETE** DELETE metod briše specificirani resurs.
- **CONNECT** Uspostavlja vezu sa serverom, onosno, identificiranim ciljanim resursom.
- **OPTIONS** OPTIONS metod opisuje komunikacione opcije za ciljani resurs.
- **TRACE** TRACE metod vrši praćenje komunikacionog linka do ciljanog resursa.
- **PATCH** PATCH metod obavlja parcijalnu modifikaciju resursa

HTTP zahtjev – **GET** metod

- GET se koristi za traženje podataka iz specificiranog izvora
- Treba imati na umu da se upitni string (par ime/vrijednost) šalje u URL-u GET zahtjeva.

/test/demo_form.php?name1=value1&name2=value2

- Nekoliko napomena u vezi GET zahtjeva:
  - GET zahtjevi se mogu keširati (spremiti u predmemoriju)
  - GET zahtjevi ostaju u istoriji pregledača
  - GET zahtjevi se mogu objeležiti
  - GET zahtjevi se nikada ne bi trebali koristiti kada se radi o osjetljivim podacima
  - GET zahtevi imaju ograničenje dužine
  - GET zahtevi se koriste samo za traženje podataka (ne promjenu)

HTTP zahtjev – **POST** metod

- POST metod se koristi za slanje podataka serveru za kreiranje/ažuriranje resursa
- Podaci poslati serveru POST metodom smješteni su u tijelu HTTP zahtjeva.

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com

name1=value1&name2=value2
```

- Nekoliko napomena u vezi POST zahtjeva:
  - POST zahtevi se nikada ne kešuju
  - POST zahtevi ne ostaju u istoriji pregledača
  - POST zahtevi se ne mogu označiti
  - POST zahtevi nemaju ograničenja u pogledu dužine podataka

HTTP zahtjev – **GET** vs. **POST** metod

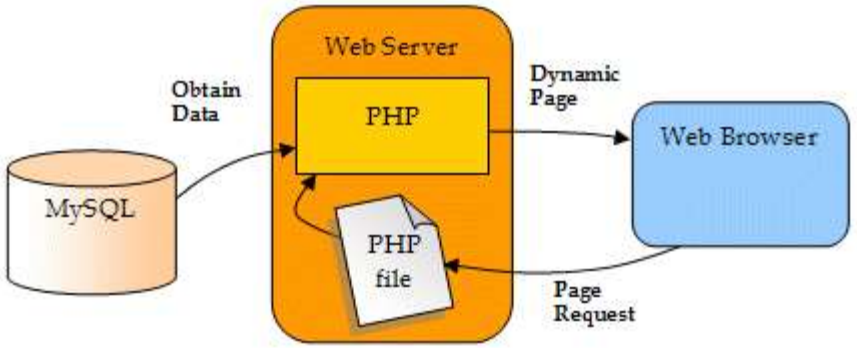
	GET	POST
BACK dugme/Ponovo učitaj	Neškodljivo	Podaci že biti ponovo poslati
Označenost	Može se označiti	Ne može se označiti
Keširanje	Može se keširati	Ne može se keširati
Tip kodiranja	application/x-www-form-urlencoded	application/x-www-form-urlencoded ili multipart/form-data. Upotreba višedjelnog kodiranja za binarne podake
Istorija	Parametri ostaju u istoriji pregledača	Parametri ne ostaju u istoriji pregledača
Ograničenja u dužini podataka	Da, kod slanja podataka, GET method dodaje podatke na URL; dužina URL-a je ograničena (maximalna URL dužina je 2048 karaktera)	Bez ograničenja
Ograničenja u tipu podataka	Samo ASCII karakteri dozvoljeni	Bez restrikcija. Binarni podaci su takođe dozvoljeni
Bezbjednost	GET manje siguran u poređenju s POST, jer su podaci dio URL-a Ne koristiti GET kada se šalje ložinka ili driuge osjetljive informacije	POST je malo sigurniji od GET jer parametri nijesu smješteni u istoriji pregledača ili u web server logu
Vidljivost	Podaci su vidljivi svima u URL-u	Podaci nijesu prikazani u URL-u

XSAMPP

- Što je XAMPP?
- XAMPP – pednosti
- Instaliranje XAMPP-a
- XAMPP – Kontrolni panel
- XAMPP – direktorijumi
- Konfigurisanje XAMPP-a
- LAMP

ŠTO JE XAMPP?

- XAMPP je besplatna open-source platforma, koja sadrži:
  - Apache HTTP server,
  - MySQL bazu podataka,
  - PHP i
  - Perl programski jezik



NAZIV XAMPP?

Naziv XAMPP je skraćenica za:

- X (čita se „kros“ i znači kros-platforma)
- Apache HTTP server
- MySQL
- PHP
- Perl

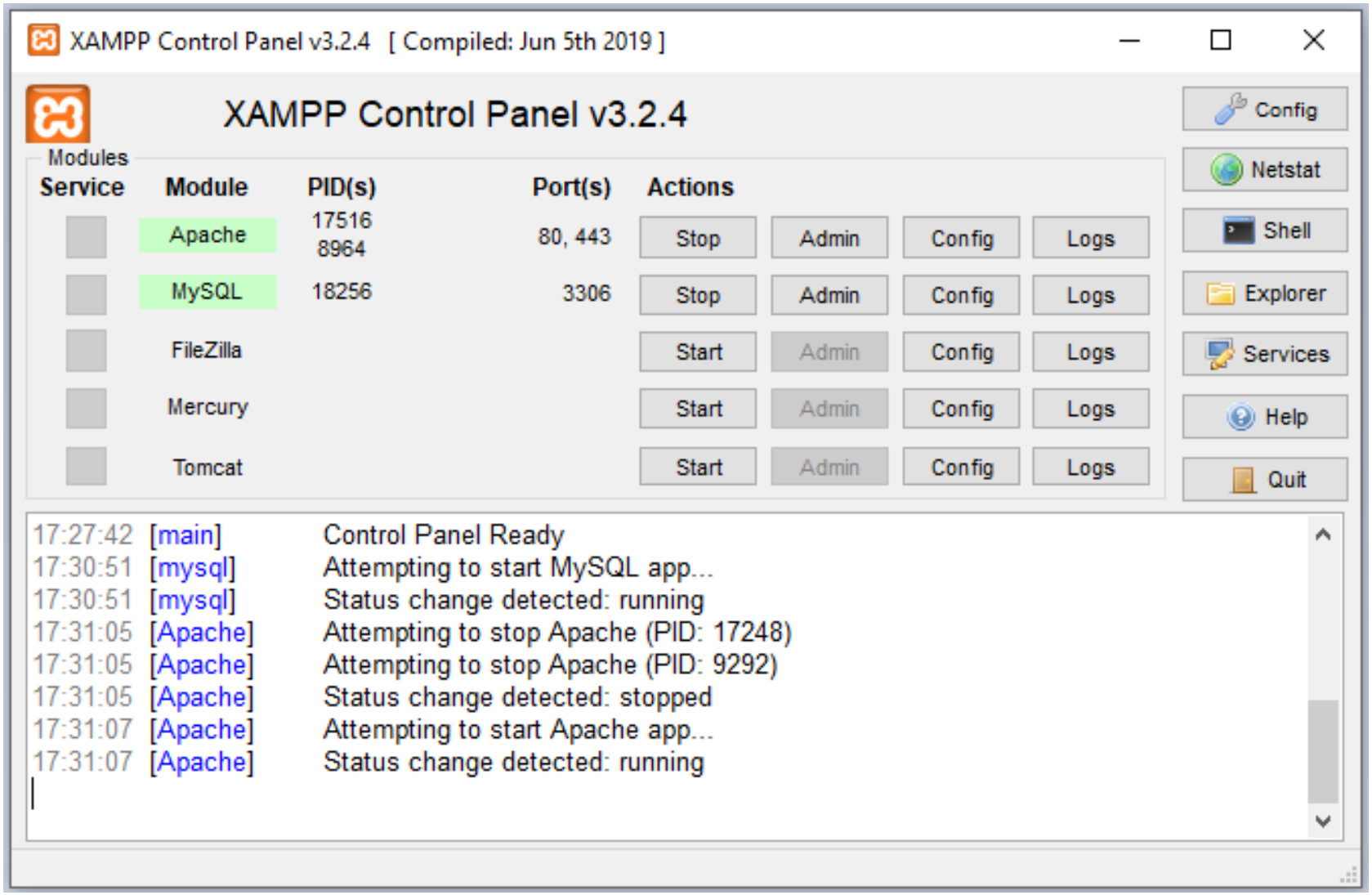
ZAŠTO KORISTITI XAMPP?

- Najpopularniji PHP razvojni paket
- Raspoloživ za Windows, Mac OS X i Linux
- Jednostavna instalacija i konfigurisanje
- Sasvim besplatan

INSTALIRANJE?

- Otići na: https://www.apachefriends.org/download.html
- Preuzeti i instalirati (može se zahtijevati pokretanja sa administratorskim pravima)
- Uobičajena windows instalacija **Next->Next-> ... -> Finish**

KONTROLNI PANEL



XAMPP DIREKTORIJUMI

- ./htdocs – lokacija javnih html fajlova
- ./apache – lokacija konfiguracija
- ./mysql – lokacija MySQL baze podataka

KONFIGURACIONI FAJLOVI

- Apache konfiguracioni fajl (httpd.conf):
  .\apache\conf\httpd.conf
- PHP konfiguracioni fajl (php.ini):
  .\apache\bin\php.ini
- MySQL konfiguracioni fajl (my.cnf):
  .\mysql\bin\mycnf

Više o instaliranju i konfigurisanju:
https://pureinfotech.com/install-xampp-windows-10/

LAMP

- **LAMP** je open-sorce Web razvojna platforma koja koristi **Linux** kao operativni sistem i **Apache** kao Web server.
- **MySQL** je upravljački sistem za relacionu bazu podataka
- **PHP** je objektno orijentisan skriptni jezik

BAZA PODATAKA

Baza podataka je skup medusobno povezanih podataka, pohranjenih bez nepotrebne redudantnosti, s ciljem da na optimalan način posluže u raznim primjenama.

- Podaci se kreiraju nezavisno od programa koji ih koriste. Zajedničkim pristupom dodaju se novi podaci, te mijenjaju i premještaju postojeći.

- Podaci se pohranjuju u bazu podataka na organizovan način, koristeći odgovarajući model podataka.

DBMS

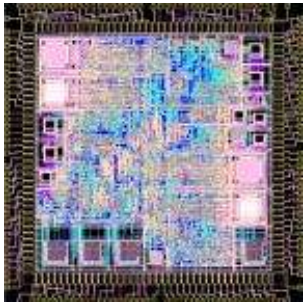
- Baza podataka je kolekcija podataka.

- Sistem za upravljanje bazom podataka (DBMS – Database Management System) je softver koji kontroliše te podatke.

DBMS

Aplikacija dolazi ovdje

DBMS



Sirovi podaci (goli metal)

DBMS interfejs omogućuje da aplikacije i sistem za upravljanje podacima budu izvedeni odvojeno

ŠTO DONOSI DBMS?

- **Obezbjeđuje:**
  - **Jezik za opis podataka** (DDL - Data definition language)
  - **Jezik za rukovanje** podacima (DML - Data manipulation language)
  - **Jezik za kontrolu** podataka (DCL - Data control language)
- **Često se ovi jezici smatraju jednim jezikom – SQL.**

- **DBMS obezbjeđuje**
  - Trajnost
  - Konkurentnost
  - Integritet
  - Bezbjednost
  - Nezavisnost podataka
- **Rječnik podataka**
  - Opisuje samu bazu podataka.

ŠTO DONOSI DBMS?

- **Fizička nezavisnost podataka.** Razdvaja se logička definicija baze od njene stvarne fizičke građe.

  Na primjer, ako se fizička građa promijeni (na primjer, podaci se prepišu u druge datoteke na drugim diskovima), to neće zahtijevati promjene u postojećim aplikacijama.

- **Logička nezavisnost podataka.** Razdvaja se globalna logička definicija cijele baze podataka od lokalne logičke definicije za jednu aplikaciju.

  Na primjer, ako se logička definicija promijeni (na primjer uvede se novi entitet ili veza), to neće zahtijevati promjene u postojećim aplikacijama.

- **Mogućnost oporavka nakon kvara.** Zaštita baze u slučaju kvara hardvera ili grešaka u radu sistemskog softvera.

ŠTO DONOSI DBMS?

- **Fleksibilnost pristupa podacima.** Korisnik može slobodno pretraživati podatke, i po želji uspostavljati veze među podacima.

  U starijim mrežnim i hijerarhijskim bazama, staze pristupanja podacima bile su unaprijed definisane. Korisnik je mogao pretraživati podatke jedino onim redoslijedom koji je bio predviđen u vrijeme projektovanja i implementiranja baze.

- **Istovremeni pristup do podataka.** Mogućnost da veći broj korisnika istovremeno koristi iste podatke.

  Korisnici ne smiju ometati jedan drugoga.

ŠTO DONOSI DBMS?

- **Zaštita od neovlašćenog korišćenja.** Mogućnost da se korisnicima ograniče prava korišćenja baze.

  Svakom korisniku se dodjeljuju ovlašćenja: što on smije, a što ne smije raditi s podacima.

- **Zadovoljavajuća brzina pristupa.** Operacije nad podacima moraju se odvijati dovoljno brzo, u skladu s potrebama određene aplikacije.

  Na brzinu pristupa može se uticati izborom pogodnih fizičkih struktura podataka, te izborom pogodnih algoritama za pretraživanje.

- **Mogućnost podešavanja i kontrole.** Velika baza zahtijeva stalnu brigu: praćenje performansi, mijenjanje parametara u fizičkoj građi, rutinsko smještanje rezervnih kopija podataka.

Danas postoji više različitih DBMS-a:

- mysql:
  - www.mysql.org
  - Open source, dosta moćan
  - MariaDB
- PostgreSQL:
  - www.postgresql.org
  - Open source, moćan
- Microsoft Access:
  - Jenostavan sistem sa puno korisnih grafičkih alata.
- Komercijalni sistemi:
  - Oracle (www.oracle.com)
  - SQL Server (www.microsoft.com/sql)
  - DB2 (www.ibm.com/db2)

Više detalja na:
https://www.ucg.ac.me/objave_spisak/blog/5742

MySQL - SQL

- SQL - Structured Query Language
- ANSI Standardi
  - SQL-1989
  - SQL-1992 (SQL2)
  - SQL-1999 (SQL3)
  - SQL-2003
  - SQL-2006
  - SQL-2008
  - SQL-2011
  - SQL-2016
  - SQL-2019
- Različiti DBMS koriste različite SQL

MySQL - SQL

- SQL obezbjeđuje
  - Jezik za opis podataka (DDL - data definition language)
  - Jezik za rukovanje podacima (DML - data manipulation language)
  - Jezik za kontrolu podataka (DCL - data control language)
- Osim toga SQL
  - se može koristiti iz drugih programskih jezika.
  - Može se proširiti u cilju obezbjeđenja uobičajenih programskih konstrukcija (kao što su: if-then, petlje, promjenljive, itd.)

MySQL - SQL

- SQL je deklarativan (neproceduralni) jezik
  - Proceduralni – navodi što kompjuter tačno treba da uradi.
  - Neproceduralni – opisuje zahtijavani rezultat (ne način kako to izračunati).

MySQL – SQL – KREIRANJE TABELE

```
CREATE TABLE
<name> (
   <col-def-1>,
   <col-def-2>,
          :
   <col-def-n>,
   <constraint-1>,
          :
   <constraint-k>)
```

- Neohodno je navesti:
  - ime tabele
  - listu definicija kolona
  - listu ograničenja (npr. ključevi)

MySQL – SQL – KREIRANJE TABELE

```
<col-name> <type>
  [NULL|NOT NULL]
  [DEFAULT <val>]
  [constraint-1 [,
   constraint-2[,
     ...]]]
```

- Svakoj koloni se zadaje ime i tip podatka koji će sadržavati
- Najčešći tipovi:
  - **INT**
  - **FLOAT**
  - **CHAR(n)**
  - **VARCHAR(n)**
  - **DATE**

MySQL – SQL – KREIRANJE TABELE

- Kolone se mogu navesti nao **NULL** ili **NOT NULL.**
- **NOT NULL** kolone ne mogu imati **NULL** vrijednost.
- Ako naredbom ništa nije navedeno za kolone, podrazumijeva se **NULL**.
- Kolonama se može dodijeliti podrazumijevena (default) vrijednost.
- Samo se navede ključna riječ DEFAULT i zatim vrijednost, primjer:

```
broj INT DEFAULT 0
```

MySQL – SQL – KREIRANJE TABELE - PRIMJER

```
CREATE TABLE Student (
   studID INT NOT NULL,
   studIme VARCHAR(50) NOT NULL,
   studAdresa VARCHAR(50),
   studGodina INT DEFAULT 1)
   ENGINE=INNODB;
```

KREIRANJE TABELE - OGRANIČENJA

CONSTRAINT

<name>

<type>

<details>

- Najčešći **<type>**:
  - **PRIMARY KEY**
  - **UNIQUE**
  - **FOREIGN KEY**
  - **INDEX**

- Ograničenja imaju ime – ograničenja pristupa zahtijevaju ime, ali neka druga ne.
- Ograničenja koja se odnose na jednu kolonu, mogu se uključiti u definiciju te kolone.

KREIRANJE TABELE - OGRANIČENJA

- Primarni ključ se definiše kroz ograničenja.
- **PRIMARY KEY** ograničenje uključuje **UNIQUE** i **NOT NULL** ograničenja.

- Primarni ključ je lista kolona koje sačinjavaju ključ.

```
CONSTRAINT <name>
   PRIMARY KEY
   (col1, col2, …)
```

KREIRANJE TABELE - OGRANIČENJA

- Isto kao **PRIMARY KEY** , grupi kolona se može zadati **UNIQUE** ograničenje
- Ovime se definiše kandidat za ključ tabele.

UNIQUE ograničenje je lista kolona koja predstavlja kandidat za ključ.

```
CONSTRAINT <name>
   UNIQUE
   (col1, col2, …)
```

OGRANIČENJA PRIMJER

```
CREATE TABLE Student (
   studID INT NOT NULL,
   studIme VARCHAR(50) NOT NULL,
   studAdresa VARCHAR(50),
   studGodina INT DEFAULT 1,
   CONSTRAINT pkStudent
     PRIMARY KEY (studID)
)ENGINE=INNODB;
```

MySQL – SQL – RUKOVANJE PODACIMA

- **`INSERT`** – dodavanje reda (zapisa) u tabelu.
- **`UPDATE`** – izmjena podataka u zapisu (zapisima) tabele
- **`DELETE`** – brisanje zapisa iz tabele

- **`UPDATE`** i **`DELETE`** koriste **`WHERE`** klauzulu kojom se specificira koje zapise izmijeniti ili ukloniti
- **BUDITE PAŽLJIVI,** netačnom **`WHERE`** klauzulom može se izgubiti puno podataka.

MySQL – SQL – INSERT

```
INSERT INTO
   <table>
   (col1, col2, …)
   VALUES
   (val1, val2, …)
```

- Broj kolona i vrijednosti mora biti isti.
- Ako se dodaju vrijednosti u svaku kolonu, ne mora se navoditi lista sa imenima kolona
- SQL ne zahtijeva da svaki zapis bude različit (osim ako neko ograničenje to ne nameće).

MySQL – SQL – INSERT - PRIMJER

Neka je polazna tabela sljedeća:

Student

ID	studIme	studAdresa	studGodina
1	Patar Marić	Slobode 23	1

```
INSERT INTO Student
(ID, studIme, studAdresa, studGodina)
VALUES (2, 'Marko Matić', 'Pobjede 12', 3)
```

Student

ID	studIme	studAdresa	studGodina
1	Patar Marić	Slobode 23	1
2	Marko Matić	Pobjede 12	3

MySQL – SQL – UPDATE

```
UPDATE <table>
SET col1 = val1
   [,col2 =
  val2…]
[WHERE
   <condition>]
```

- U svim vrstama kod kojih je uslov zadovoljen postavljaju se zadate vrijednosti kolonama.
- BUDITE PAŽLJIVI - ako nije zadat uslov svi zapisi će biti promijenjeni.
- Vrijednosti su konstante ili algebarski izrazi.

MySQL – SQL – UPDATE - PRIMJER

```
UPDATE Student
SET studGodina = 2, studIme = 'Marina Šoć'
WHERE ID = 4
```

Student

ID	studIme	studAdresa	studGodina
1	Patar Marić	Slobode 23	1
2	Marko Matić	Pobjede 12	3
3	Ana Tot		1
4	Marina Šoć	Cetinjski put bb	2

```
UPDATE Student
SET studGodina = studGodina + 1
```

Student

ID	studIme	studAdresa	studGodina
1	Patar Marić	Slobode 23	2
2	Marko Matić	Pobjede 12	4
3	Ana Tot		2
4	Marina Šoć	Cetinjski put bb	3

MySQL – SQL – DELETE

- Ukljanja sve zapise koji zadovoljavaju uslov

```
DELETE FROM
   <table>
    [WHERE
     <condition>]
```

- Ako neme uslova, onda će SVI zapisi biti obrisani – BUDITE PAŽLJIVI!
- Neke verzije SQL-a imaju i naredbu `TRUNCATE TABLE <T>` koja je kao i `DELETE FROM <T>` ali je u nekim situacijama brža.

MySQL – SQL – DELETE - PRIMJER

```
DELETE FROM
Student
WHERE studGodina = 2
```

Student

ID	studIme	studAdresa	studGodina
2	Marko Matić	Pobjede 12	4
4	Marina Šoć	Cetinjski put bb	3

Student

ID	studIme	studAdresa	studGodina

MySQL – SQL – SELECT

- SQL komanda koja se najčešće koristi.
  - Upit prema grupi tabela. Rezultat je takođe tabela.
  - Puno opcija.
  - Obično postoji više načina za sastaviti bilo koji upit.

```
SELECT
   [DISTINCT | ALL] <column-list>
   FROM <table-names>
   [WHERE <condition>]
   [ORDER BY <column-list>]
   [GROUP BY <column-list>]
   [HAVING <condition>]
```

- *([ ] - optional, | - or)*

Više detalja na:
https://www.ucg.ac.me/objave_spisak/blog/5742

PHPMyAdmin

- Web aplikacija
- Olakšava upotrebu MySQL baze podataka (grafički interfejs)
- Pokreće se na:

  http://localhost/phpmyadmin

- Preuzima se na:

  http://www.phpmyadmin.net/home_page/downloads.php

Kako koristiti PHPMyAdmin (u više detalja):
https://www2.slideshare.net/karwanmst/mysql-database-with-phpmyadmin?from_action=save

MySQL – SQL – NAŠA BAZA

Naziv baze: idsys

Tabele:

- orgjedinice – organizacione jedinice korisnika
- korisnici – korisnici čiji identifikatori ostvaruju pravo pristupa
- terminali – uređaji na kojima se očitavaju identifikatori
- evidencije – evidentiranje očitanja identifikatora

MySQL – SQL – TABELA KORISNICI

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**IDKorisnika**	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**Prezime**	varchar(25)	utf8mb4_general_ci		Yes	*NULL*		
☐	3	**Ime**	varchar(25)	utf8mb4_general_ci		Yes	*NULL*		
☐	4	**TagID**	varchar(16)	utf8mb4_general_ci		No	*None*		
☐	5	**IDOrgJed**	int(11)			No	*None*		
☐	6	**Aktivan**	tinyint(1)			No	*None*		

MySQL – SQL – TABELA ORGJEDINICE

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**IDOrgJed** 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**NazivOrgJed**	varchar(100)	utf8mb4_general_ci		Yes	*NULL*		
☐	3	**IDRod**	int(11)			Yes	*NULL*		

MySQL – SQL – TABELA EVIDENCIJE

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	ID 🔑	int(11)			No	*None*		AUTO_INCREMENT
☐	2	IDKorisnika	int(11)			No	*None*		
☐	3	Vrijeme	datetime			No	*None*		
☐	4	TerminalID	int(11)			No	*None*		
☐	5	tagID	varchar(16)	utf8mb4_general_ci		No	*None*		

MySQL – SQL – TABELA TERMINALI

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**TerminalID**	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**Adresa**	int(11)			No	*None*		
☐	3	**Naziv**	varchar(100)	utf8mb4_general_ci		Yes	*NULL*		
☐	4	**Tip**	int(11)			Yes	*NULL*		

PHP

https://www.w3schools.com/php/

ZA VJEŽBU

Doraditi arduino skeč i PHP fajl tako da se omogući sljedeće:

- Nakon očitavanja MASTER kartice uređaj ulazi u mod upisivanja i brisanja korisnika.
- Nakon primicanja nepoznatog tag-a treba napraviti da se upiše novi red u tabeli korisnici u kojem će se u koloni TagID upisati ID upravo očitanog taga, u koloni TerminalID adresa uređaja, a u koloni Aktivan broj 1. U kolonama Ime, Prezime i IDOrgJed upisati vrijednost NULL. Kolona IDKorisnika je AUTO_INCREMENT i u njoj ne treba upisivati vrijednost kroz PHP program.
  (2 -1 bod)
- Nakon primicanja poznatog tag-a treba napraviti da se obriše red iz tabele korisnici u kojem je u koloni TagID upisan ID upravo očitanog taga.
  (2-1 bod)
- Ponovnim primicanjem MASTER kartice vraća se u mod prepoznavanja.

- Kada je u modu prepoznavanja, treba da radi na sljedeći način:
- Nakon primicanja tag-a treba napraviti da se najprije u tabeli korisnici provjeri da li postoji red u kojem je u koloni TagID upisan ID upravo očitanog taga.
  - Ukoliko postoji, uzeti IDKorisnika čiji je to TagID. U tabeli evidencije upisati novi red, u kojem će se u koloni TagID upisati ID upravo očitanog taga, u koloni TerminalID adresa uređaja, u koloni IDKorisnika broj preuzet iz tabele korisnici, dok u koloni vrijeme treba upisati trenutni datum i vrijeme.
  - Ukoliko ne postoji, u tabeli evidencije upisati novi red, u kojem će se u koloni TagID upisati ID upravo očitanog taga, u koloni TerminalID adresa uređaja, u koloni IDKorisnika broj -1, dok u koloni vrijeme treba upisati trenutni datum i vrijeme.

(2-1 bod)

Osvojeni bodovi se sabiraju