Priprema za završni pisani ispit

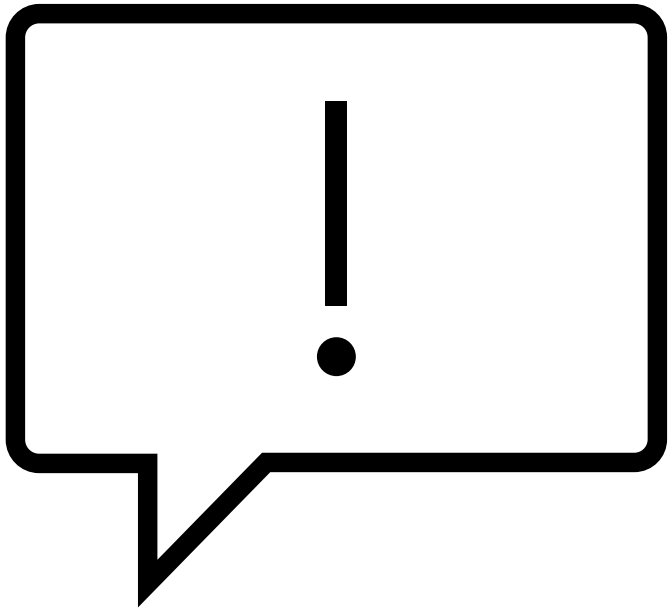
Back-End developer na PHP-u

Što sadrži termin pripreme za ispit?

- Radno okruženje
- PHP osnove
- MySQL osnove
- Napredno PHP programiranje
- Radni okvir – Laravel
- Prijenos podataka na produkcijsko okruženje

Važno!

- Prezentacija pripreme za završni ispit strukturirana je prema modulima/ seminarima koji prate tijek nastavnog procesa te obuhvaća teoriju i zadatke
- Prezentacija obuhvaća **ključan sadržaj** čija se razina usvojenosti može propitati pismenom provjerom znanja putem ispitne aplikacije MyQtest
- Cilj je kroz prezentaciju ponoviti već obrađeno gradivo, a ne obrađivati nove teme / pojmove
- Zadaci u samom ispitu **podložni su promjenama** kako bi korespondirali s aktualnim trendovima u stvarnom poslovnom sektoru



Tijekom pripreme za ispit važno je služiti se svim dostupnim nastavnim materijalima!

Uređivanje radnog okruženja za rad u PHP-u

Radno okruženje

- Radno okruženje je skup alata, programa i postavki koje se koriste za obavljanje određenog posla ili zadatka na računalu.
- To uključuje operativni sustav, softverske aplikacije, programerske alate, konfiguracijske datoteke, skripte i sve druge alate potrebne za obavljanje posla.

Izvor: Freepik Storyset

Neke od osnovnih Linux naredbi i njihovo značenje

ls - listanje sadržaja trenutnog direktorija

cd - promjena trenutnog direktorija

pwd - ispisuje putanju do trenutnog direktorija

mkdir - stvaranje novog direktorija

rmdir - brisanje direktorija

touch - stvaranje novih datoteka

rm - brisanje datoteka

cp - kopiranje datoteka

Neke od osnovnih Linux naredbi i njihovo značenje

mv - premještanje datoteka

cat – ispisivanje i pregled sadržaja datoteke

less - pregledavanje datoteke koristeći manje memorije

grep - pretraživanje datoteke za određenim nizom znakova

top - ispisivanje popisa procesa koji trenutno rade

ps - ispisivanje popisa procesa koje je pokrenuo trenutni korisnik

chmod - promjena dozvola pristupa datotekama i direktorijima

Vl editor

- Vl je tekstualni *editor* koji se često koristi u Unix i Linux sustavima.
- Vl je kratica za ***visual editor***.
- Vl *editor* ima dva načina rada: **naredbeni način** i **način umetanja**.
- Vl posjeduje UNDO mogućnost.

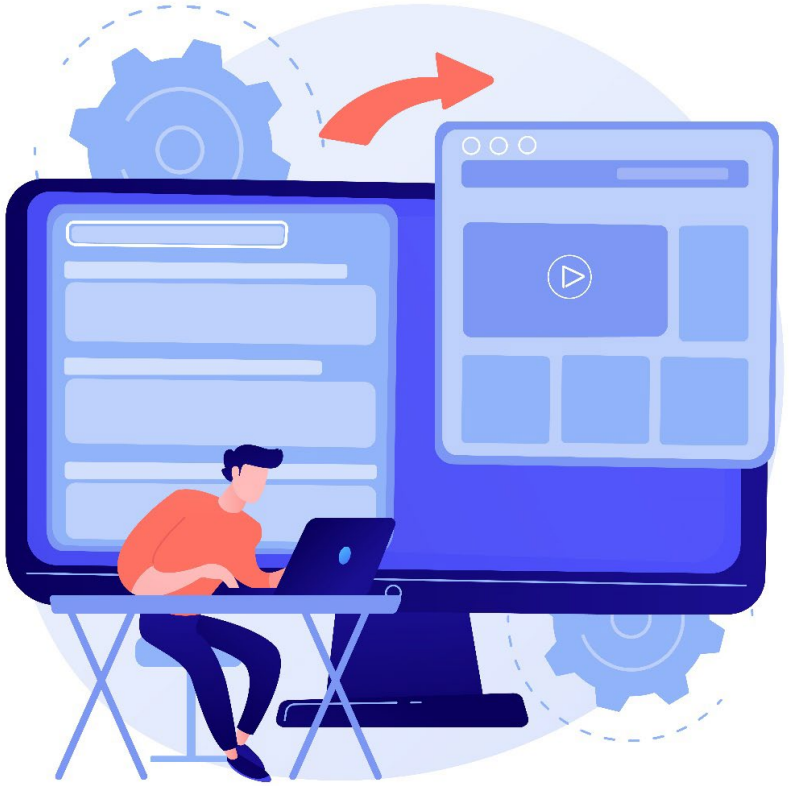
Koje su prednosti, a koji nedostaci VI editora?

Neke od osnovnih naredbi VI editora

i - ulazak u način rada za uređivanje teksta	**ESC** - izlazak iz načina rada za uređivanje teksta	**:w** - spremanje datoteke
:q - izlazak iz VI editora	**:wq** - spremanje datoteke i izlazak iz VI editora	**:q!** - izlazak iz VI editora bez spremanja promjena

Koja je svrha kontrole verzija u razvoju softvera?

- Svrha kontrole verzija u razvoju softvera je ***praćenje promjena*** u izvornom kodu softvera tijekom vremena, kako bi se omogućilo ***lakše upravljanje timom developera***, ***praćenje promjena*** i ***vraćanje na prethodne verzije softvera*** ako je potrebno.
- Git je distribuirani sustav za upravljanje verzijama softvera koji se koristi za praćenje promjena u izvornom kodu tijekom vremena.



Izvor: Freepik Storyset

Neke od osnovnih naredbi Git-a

git init : Inicijalizira novi Git repozitorij u lokalnom direktoriju.

git add : Dodaje datoteke u indeks za sljedeći commit.

git commit : Stvara commit s trenutnim indeksom i otvara prozor za unos poruke.

git status : Prikazuje stanje radnog direktorija i indeksa.

git log : Prikazuje listu commitova u repozitoriju.

git branch : Prikazuje popis lokalnih grana i stvara novu granu.

git checkout : Prebacuje trenutni direktorij na drugu granu ili commit.

git merge : Spaja dvije grane ili commita.

git clone : Klonira udaljeni repozitorij na lokalno računalo.

git pull : Povlači najnovije promjene sa udaljenog repozitorija i spaja ih s lokalnom verzijom.

git push : Šalje lokalne promjene na udaljeni repozitorij.

Savjeti za učinkovit rad s Git-om!

- ✓ Redovito commitajte svoje promjene: Dobro je napraviti commit nakon svake značajnije promjene u kodu. Ovo čini povijest vašeg koda jasnijom i olakšava praćenje promjena.
- ✓ Komentirajte svoje commit poruke: U svojoj commit poruci uvijek napišite kratak opis promjena koje ste napravili. To će pomoći drugim programerima da shvate što ste promijenili i zašto ste to učinili.
- ✓ Koristite grane: Grane (branches) su odličan način da eksperimentirate s novim značajkama i da radite na više stvari istovremeno. Ako radite na nečemu što bi moglo pokvariti stabilnost glavne grane, možete raditi na svojoj grani sve dok niste spremni za spajanje (merge).
- ✓ Redovito ažurirajte svoj repozitorij: Povlačenje (pull) promjena s drugih grana i repozitorija važno je da bi se osiguralo da imate najnoviju verziju koda s kojim radite.
- ✓ Upotrijebite .gitignore: Ako imate datoteke koje ne želite pratiti s Git-om (npr. privremene datoteke, lozinke i sl.), dodajte ih u .gitignore datoteku kako bi Git ignorirao ove datoteke.
- ✓ Usvojite osnovne koncepte Git-a: Da bi ste bolje razumjeli Git, trebali bi se upoznati s osnovnim konceptima kao što su indeks (index), grane (branches), commiti i sl.
- ✓ Redovito back-up-irajte vaše datoteke: Iako Git čuva sve promjene u kodu, dobro je imati back-up vašeg repozitorija u slučaju gubitka podataka ili kvara hardvera.

Što je Apache?

- Apache je popularni web server softver, koji se koristi za hostanje web stranica i aplikacija. Pokrećemo ga sa **”start httpd”**
- Konfiguracija **httpd.conf** Apache-a se uobičajeno nalazi u direktoriju /etc/apache2/
- Zadani korijenski (root) web direktorij za Apache web poslužitelj na Linuxu je **/var/www/html**
- Direktiva koja se koristi u datoteci **httpd.conf** za određivanje korijenskog web direktorija je **DocumentRoot**
- Direktiva koja se koristi u datoteci **httpd.conf** za određivanje zadane (početne) datoteke za posluživanje je **DirectoryIndex**
- Za verzije Apache poslužitelja 2.4 i novije, naredba za provjeru sintakse konfiguracijske datoteke je: **apachectl configtest**

Što je MySQL?

- MySQL je popularni besplatni relacijski sustav za upravljanje bazama podataka (RDBMS) koji se koristi za pohranu i upravljanje podacima u aplikacijama.
- Pokrećemo ga sa **”start mysqld”**
- Naredba za prikaz svih baza podataka u MySQL-u je **„SHOW DATABASES”.**

Što je PHP?

- PHP (*Hypertext Preprocessor*) je popularni *open-source* programski jezik koji se koristi za razvoj web aplikacija.
- PHP se obično izvršava na web poslužiteljima, a kôd PHP-a koristi se za generiranje dinamičkih web stranica koje se prikazuju u pregledniku.
- Konfiguracijska datoteka za PHP je "**php.ini**"

Što je Composer?

- Composer je alat za upravljanje ovisnostima u PHP-u.
- Omogućuje programerima da lako dodaju i uklanjaju pakete (biblioteke, alate i sl.) u svoj projekt, zatim automatski riješi ovisnosti između tih paketa.
- Instalacija i konfiguracija:
  1. Preuzmite i instalirajte Composer prema uputama s njegove službene stranice (**https://getcomposer.org/**).
  2. Konfiguracija se vrši uređivanjem **"composer.json"** datoteke, koja se nalazi u korijenskom direktoriju vašeg projekta.

Instaliranje specifične verzije paketa

- Da biste instalirali specifičnu verziju paketa, možete to navesti u "**composer.json**" datoteci u sekciji "**require**", npr.:

Ponovimo

Koja se naredba koristi za stvaranje novog direktorija u Linuxu?

A) mkdir
B) cd
C) ls
D) touch

Ponovimo

Koja je naredba za pokretanje MySQL poslužitelja?

A) start mysqld
B) start mysql
C) start sqlserver
D) start server

Ponovimo

Kako možete instalirati određenu verziju paketa koristeći Composer?

A) Ručnim uređivanjem datoteke composer.lock.
B) Pokretanjem naredbe "composer install" s brojem verzije kao argumentom.
C) Navođenjem broja verzije u datoteci composer.json.
D) Pokretanjem naredbe "composer require" s nazivom paketa i brojem verzije kao argumentima.

Zadatak

Što trebate učiniti da biste instalirali verziju "1.0.0" paketa "vendor/package" pomoću Composer-a?

1. Otvorite naredbeni redak (command prompt) na vašem računalu.
2. Pomaknite se do direktorija u kojem želite instalirati paket "vendor/package".
3. Izvršite sljedeću naredbu:
   composer require vendor/package:1.0.0

Osnove PHP-a

Što je to HTML?

- **HTML (*HyperText Markup Language*)** je standardni jezik za izgradnju web stranica. HTML se koristi za definiranje strukture, sadržaja i izgleda web stranica. To je osnovni jezik koji se koristi za kreiranje svih web stranica na Internetu.
- Zadnja verzija HTML-a je **HTML5**, koja je objavljena 2014. godine.

Izvor: Free Pik

Tagovi, atributi?

- HTML tagovi su elementi koji se koriste za strukturiranje i oblikovanje sadržaja na web stranici
- Atributi su dodatne informacije koje se mogu dodati tagovima kako bi se pružile dodatne informacije o elementu ili kako bi se prilagodile njegove postavke.
- Prvi red u vašem HTML kodu bi trebao biti **<!DOCTYPE html>**
- Atribut koji se koristi za definiranje kodiranja znakova za HTML document je <meta **charset**="UTF-8">.
- Vizualni stilovi se definiraju uz pomoć CSS-a
- Povezivanje vanjskim stilom se vrši na sljedeći način: **<link rel="stylesheet" href="style.css">**

Neke od osnovnih HTML naredbi

<html> - označava početak HTML dokumenta.

<head> - označava početak zaglavlja HTML dokumenta i obično sadrži informacije o dokumentu, poput naslova stranice, meta podataka i skripti.

<title> - označava naslov HTML dokumenta i pojavljuje se u okviru <head> taga.

<body> - označava početak tijela HTML dokumenta i sadrži sve vizualne elemente na stranici.

<h1> - <h6> - označavaju naslove različitih razina (od najvažnijeg do najmanje važnog).

Neke od osnovnih HTML naredbi

<p> - označava novi paragraf.

<a> - označava hipervezu i koristi se za stvaranje linkova na druge stranice ili dokumente.

<img> - označava sliku i koristi se za prikazivanje slika na web stranicama.

<ul> i **<ol>** - označavaju neuređenu (**ul**) i uređenu (**ol**) listu.

<li> - označava stavke u listi.

<table> - označava tablicu i koristi se za prikazivanje strukturiranih podataka u obliku tablice.

PHP - sažetak

- Skriptni jezik na strani poslužitelja
- Sintaksa za pokretanje PHP skripte je **<?php**
- Za ispis teksta na ekran koristimo naredbu **„echo**”
- PHP komentari: **#**, **/* */**, **//**

Izvor: Free Pik

Prisjetite se: Koje tipove podataka poznajete?

string	*float*	*boolean*
integer	*object*	*array*
	NULL	

PHP- sažetak

- **"=="** uspoređuje samo vrijednosti, dok **"==="** uspoređuje i vrijednosti i tipove podataka.
- **INCLUDE** generira samo upozorenje ako datoteka nije pronađena, dok **REQUIRE** generira fatalnu pogrešku.
- Svrha PHP funkcije "**spl_autoload_register**„ je registrirati funkciju koja se automatski poziva kada je klasa nedefinirana.
- Imenski prostor (**namespace**) u PHP-u je način izbjegavanja sukoba naziva između klasa, funkcija i varijabli.

Primjer

PHP kod koji koristi ***foreach petlju*** za prolazak kroz sve elemente polja i ispisivanje njihove vrijednosti

```php
<?php
$polje = array("jabuka", "banana",
"naranča");

foreach ($polje as $vrijednost) {
    echo "$vrijednost <br>";
}
?>
```

Ponovimo

Koja je najnovija verzija HTML-a?

A) HTML 4.01
B) XHTML
C) HTML5
D) HTML 3.2

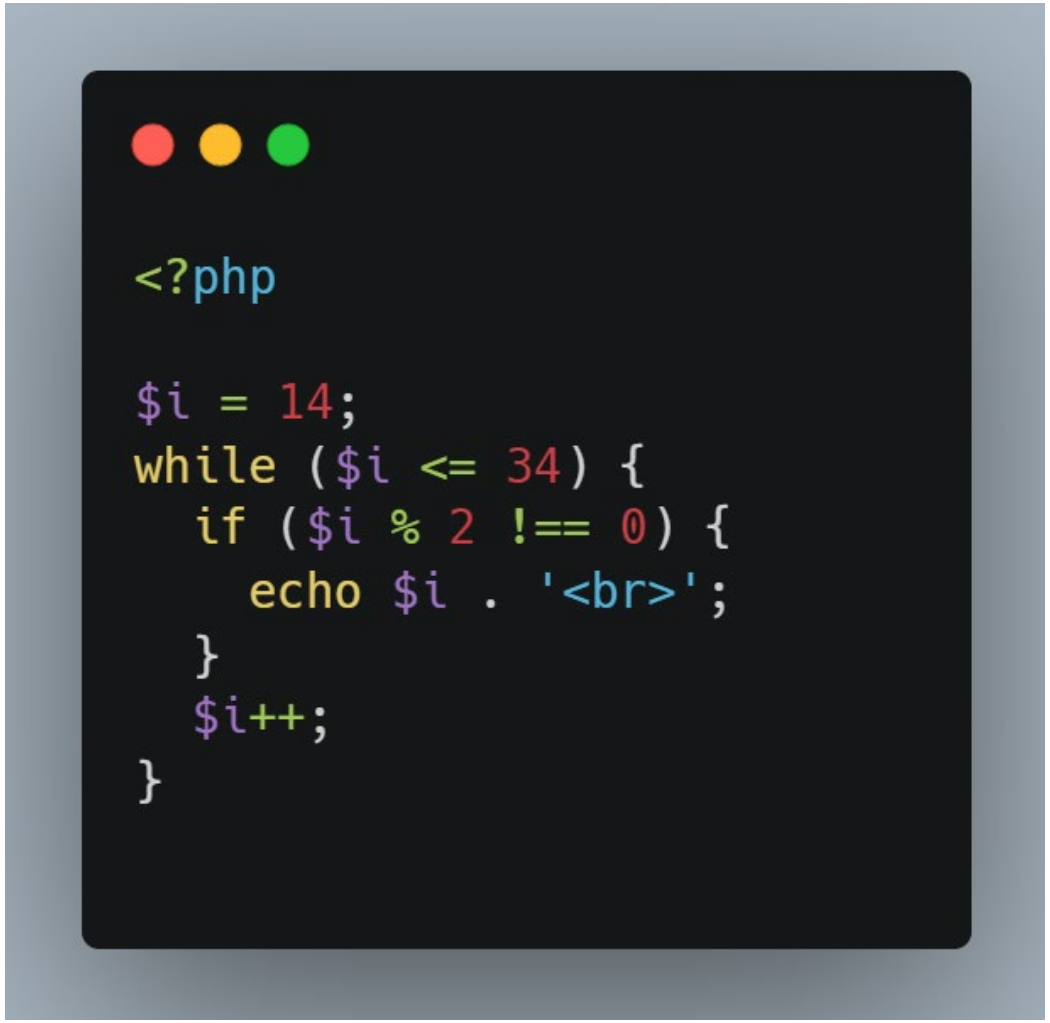
Zadatak 1

Napišite kratak PHP kod koji koristi **Do-While** petlju za ispisivanje brojeva od 49 do 13.

```php
<?php

$broj = 49;
do {
  echo $broj . '<br>';
  $broj--;
} while ($broj >= 13);
```

Zadatak 2

Napišite kratki PHP kod koji koristi **While** petlju za ispisivanje neparnih brojeva od 14 do 34.



```php
<?php

$i = 14;
while ($i <= 34) {
  if ($i % 2 !== 0) {
    echo $i . '<br>';
  }
  $i++;
}
```

Zadatak 3

Napišite kratki PHP kod koji koristi **for** petlju za ispis svih brojeva od 1-100 koji su djeljivi sa 7 ili 9 ali preskoči brojeve 63, 70 i 90.

```php
<?php

//Primjer:
for ($i = 1; $i <= 100; $i++) {
  if (($i % 7 == 0 || $i % 9 == 0)
&& $i != 63 && $i != 70 && $i != 90)
{
    echo $i . '<br>';
  }
}

// primjer uz pomoć "continue"
for ($i = 1; $i <= 100; $i++) {
  if ($i == 63 || $i == 70 || $i ==
90) {
    continue;
  }
  if ($i % 7 == 0 || $i % 9 == 0) {
    echo $i . '<br>';
  }
}
```

PHP varijable

- **Varijable** se koriste za pohranu podataka koji se kasnije mogu koristiti tijekom izvršavanja programa.
- **Globalne** varijable su definirane izvan funkcije i mogu se koristiti unutar bilo koje funkcije. One su dostupne u cijelom programu. Deklariranje globalne varijable koristimo **$GLOBALS['variable_name'];**
- **Static** varijable su lokalne varijable, ali zadržavaju svoju vrijednost između poziva funkcije.

```php
<?php

$globalna_varijabla = "Ovo je
globalna varijabla!";

function ispisi_globalnu_varijablu()
{
    global $globalna_varijabla;
    echo $globalna_varijabla;
}
// Ispisuje
// Ovo je globalna varijabla!"
```

PHP funkcije

- U PHP-u, funkcija je blok koda koji se može pozivati iz drugih dijelova programa kako bi se izvršile određene radnje.
- Primjer jednostavne funkcije koja ispisuje pozdravni tekst: ***function*** *pozdrav($ime) { echo "Pozdrav, $ime!"}*
- Kako bi vratili vrijednost iz PHP funkcije koristimo **return $value**
- PHP funkcija koja prihvaća promjenjivi broj argumenata: *function naziv_funkcije(**...$args**) { // tijelo funkcije }*
- Kada proslijedite argument **referencom** na funkciju u PHP-u tada funkcijom **modificiramo izvornu varijablu**.

Nizovi u PHP-u

- Niz se stvara korištenjem funkcije array() ili skraćenog zapisa [].
- Indeksirani nizovi koriste cijele brojeve kao ključeve, dok asocijativni nizovi koriste tekstovne znakove kao ključeve.
- Primjer: **$boje = array("crvena", "plava", "zelena"); echo $boje[0]; // ispisuje "crvena“**

- Neke od čestih naredbi za rad s nizovima u PHP-u:
  - **count()** - vraća broj elemenata u nizu
  - **sort()** - sortira niz u rastućem redoslijedu
  - **rsort()** - sortira niz u padajućem redoslijedu
  - **array_push()** - dodaje novi element na kraj niza
  - **array_pop()** - uklanja zadnji element niza i vraća ga kao vrijednost

Zadatak

Napišimo funkciju u PHP-u koja prima **dva parametra:**

- broj brojeva koji se generiraju i s kojim brojem bi trebali biti djeljivi
- dodaje generirane brojeve na kraj polja.

```php
<?php

function generiraj_djeljive_brojeve($broj_brojeva,
$djeljiv_s, &$niz_brojeva) {
  for ($i = 1; $i <= $broj_brojeva; $i++) {
      array_push($niz_brojeva, $i*$djeljiv_s);
  }
  return $niz_brojeva;
}

$niz_brojeva = array(10, 20, 30, 40, 50);
$djeljiv_s = 7;
$broj_brojeva = 5;

generiraj_djeljive_brojeve($broj_brojeva,
$djeljiv_s, $niz_brojeva);
print_r($niz_brojeva);

/*
Array
(
    [0] => 10
    [1] => 20
    [2] => 30
    [3] => 40
    [4] => 50
    [5] => 7
    [6] => 14
    [7] => 21
    [8] => 28
    [9] => 35
)
*/
```

PHP Cookies

- **Cookies** su male tekstualne datoteke koje se pohranjuju na računalu korisnika kada korisnik posjeti web stranicu.
- Za postavljanje cookiesa u PHP-u koristi se funkcija **setcookie()**
- Za čitanje vrijednosti cookie-a u PHP-u koristi se globalna varijabla $_COOKIE, npr:
  echo $_COOKIE['username’];
- Za brisanje cookie-a u PHP-u treba postaviti vrijeme isteka na vrijeme koje je prošlo npr:
  setcookie(„Korisnicki_broj", "", time()-3600, "/");

PHP - pisanje i čitanje datoteka

- **fopen():** otvara datoteku u određenom načinu (npr. čitanje, pisanje, dodavanje) i vraća pokazivač datoteke.
  - Primjer**:** **$handle = fopen('datoteka.txt', 'r');**

- **fwrite():** piše u datoteku s otvorenim pokazivačem datoteke.
  - Primjer: **fwrite($handle, 'Ovo je primjer teksta koji se piše u datoteku.');**

- **fclose():** zatvara datoteku s otvorenim pokazivačem datoteke.
  - Primjer: **fclose($handle);**

PHP - pisanje i čitanje datoteka

- **unlink():** briše datoteku na navedenoj putanji.
  - Primjer: **unlink('datoteka.txt');**

- Je li datoteka učitana s forme provjeravamo s **$_FILES["datoteka"]**

- Za premještanje datoteke koristimo funkciju **move_uploaded_file()**

Pitanja za ponavljanje

1. Koja se oznaka koristi za izradu tablice u HTML-u?
2. Koji se atribut koristi za definiranje ciljanog URL-a za hipervezu u HTML-u?
3. Koje sve PHP razvojne okvire (framework) poznajete?
4. Koja je svrha PHP funkcije “echo”?
5. Koja je razlika između "==" i "===" u PHP-u?
6. Koja je razlika između null vrijednosti i praznog stringa u PHP-u?
7. Koja je sintaksa za deklariranje funkcije sa zadanom vrijednošću parametra u PHP-u? *(zadatak)*
8. Kako deklarirati PHP funkciju koja prihvaća promjenjivi broj argumenata? *(zadatak)*
9. Kako zatvoriti datoteku u PHP-u?
10. Kako se stvara niz u PHP-u?

Osnove MySQL-a

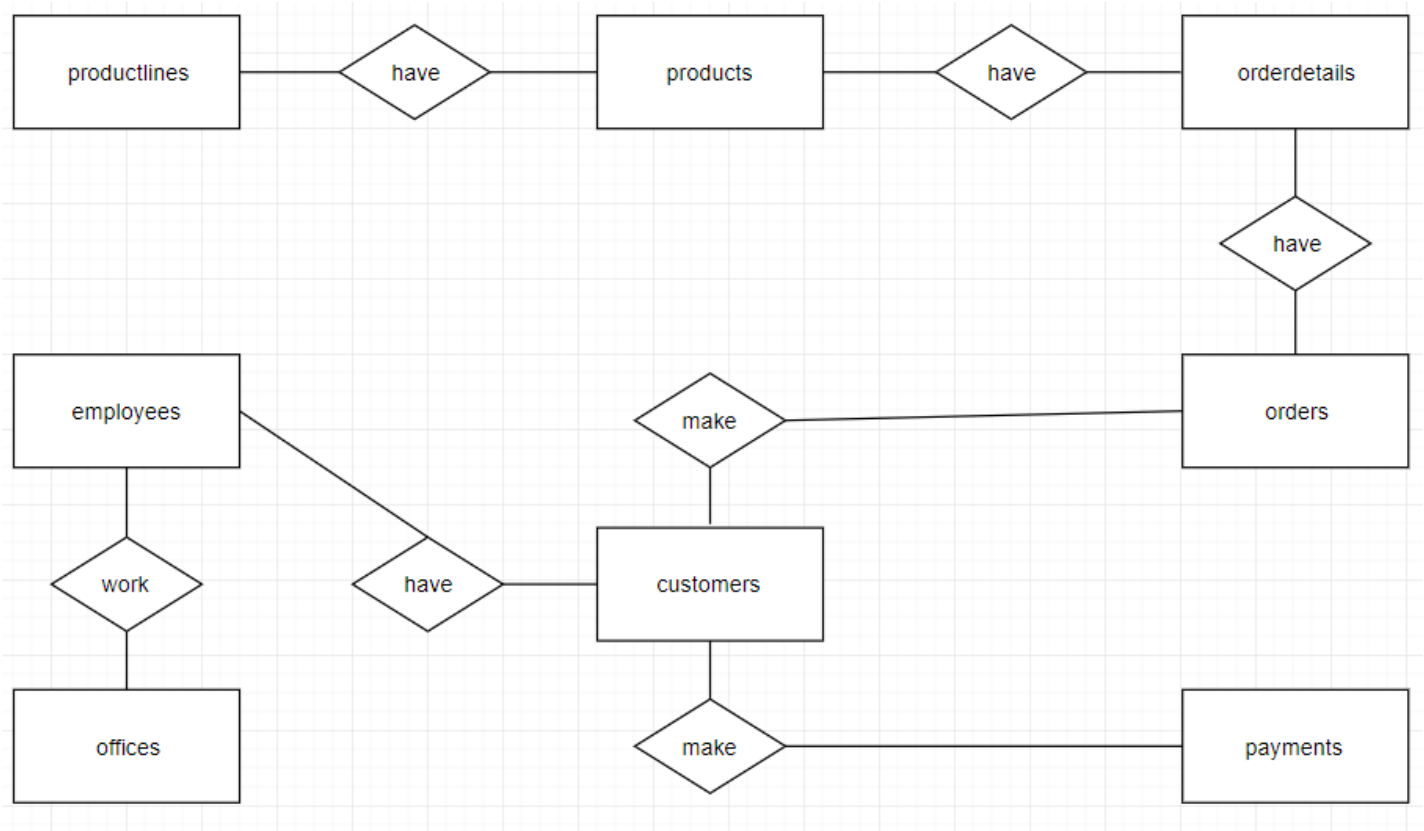
Osnove MySQL-a

- MySQL je besplatni *open-source* sustav za upravljanje relacijskim bazama podataka.
- ER (*Entity-Relationship*) dijagram je grafički prikaz veza između entiteta u bazi podataka.
- Ovaj dijagram prikazuje shemu baze podataka i pomaže u vizualnom predstavljanju relacija između različitih tablica i entiteta.
- ER dijagrami se sastoje od entiteta, veza i atributa.
  - Entiteti predstavljaju različite tablice ili objekte u bazi podataka.
  - Veze prikazuju relacije između različitih entiteta.
  - Atributi su svojstva entiteta i mogu biti primarni ključevi, strani ključevi ili jednostavni atributi.

Simboli u ER dijagramu uključuju:

- **Entitet**: Predstavljen **pravokutnikom** s imenom entiteta unutar njega.
- **Atribut**: Predstavljen **elipsom** s imenom atributa unutar njega. Može biti primarni ključ, strani ključ ili jednostavni atribut.
- **Veza**: Predstavljen **linijom** koja povezuje dva ili više entiteta. Može biti jedan-na-jedan, jedan-na-mnogo, mnogo-na-jedan ili mnogo-na-mnogo.
- **Strani ključ:** Identifikator entiteta koji se koristi za stvaranje veze s drugom tablicom. Označen je **rombom**.
- **Kardinalnost**: Ovaj simbol označava broj entiteta koji se mogu povezati preko veze. Može biti 1 (jedan), N (mnogi) ili 0..1 (nijedan ili jedan).

ER dijagram


productlines
have
products
have
orderdetails
have
employees
make
orders
work
have
customers
offices
make
payments

Normalizacija

- U normalizaciji baze podataka, forme predstavljaju različite normalne forme koje baza podataka treba ispuniti kako bi se postigla efikasnost, fleksibilnost i integritet podataka.
- Tablica je u **1NF** ako nema ponavljajućih grupa ili nizova.
- **Druga normalna forma (2NF):** Svaka tablica mora imati primarni ključ, a svi ostali atributi u tablici moraju biti ovisni o primarnom ključu.
- **Treća normalna forma (3NF):** Svaki atribut koji nije dio primarnog ključa treba biti ovisan samo o primarnom ključu, a ne na drugim atributima u tablici

MySQL naredbe

DLM naredbe

Data Manipulation Language

Koriste se za manipulaciju podacima u tablicama.

DCL naredbe

Data Control Language

Skup naredbi u MySQL-u koje se koriste za kontrolu pristupa podacima i upravljanje sigurnošću sustava.

DDL naredbe

Data Definition Language

Koriste se za definiranje, mijenjanje i brisanje struktura baze podataka i njenih objekata. Naredbe se koriste za stvaranje tablica, indeksa, ograničenja, gledišta i drugih objekata baze podataka.

MySQL DML naredbe

- **SELECT** - koristi se za dohvaćanje podataka iz tablice ili više tablica.
- **INSERT** - koristi se za umetanje novih redova u tablicu.
- **UPDATE** - koristi se za ažuriranje postojećih zapisa u tablici.
- **DELETE** - koristi se za brisanje redova iz tablice.

Razmislite

Kako bi ste uz pomoć DML naredbi opisali kupovinu cipela? (CRUD)

```sql
-- CREATE
INSERT INTO shopping (item, price,
date)
VALUES ('cipele', 100, '2023-04-30');

-- READ
SELECT item, price, date
FROM shopping
WHERE item = 'cipele'
ORDER BY price DESC
LIMIT 5;

-- UPDATE
UPDATE shopping
SET price = 120, date = '2023-05-01'
WHERE item = 'cipele';

-- DELETE
DELETE FROM shopping
WHERE item = 'cipele';
```

MySQL DCL naredbe

- **GRANT** - dodjeljuje korisniku pravo pristupa bazi podataka i tablicama. Ova naredba može se koristiti za određivanje prava pristupa i ograničenja na razini korisnika ili na razini baze podataka.
- **REVOKE** - uklanja pravo pristupa koje je dodijeljeno korisniku pomoću GRANT naredbe.
- **CREATE USER** - stvara novog korisnika i dodjeljuje mu lozinku za pristup MySQL bazi podataka.
- **DROP USER** - briše korisnika iz MySQL baze podataka.
- **ALTER USER** - mijenja korisničko ime ili lozinku postojećeg korisnika.
- **CREATE ROLE** - stvara novu ulogu koja može biti dodijeljena korisniku.
- **DROP ROLE** - briše ulogu iz MySQL baze podataka.

MySQL DDL naredbe

- **CREATE**: koristi se za stvaranje novih objekata baze podataka, kao što su tablice, indeksi ili ograničenja.
- **ALTER**: koristi se za promjenu strukture postojećih objekata baze podataka, kao što su dodavanje ili brisanje stupaca, indeksa ili ograničenja.
- **DROP**: koristi se za brisanje objekata baze podataka, kao što su tablice, indeksi ili ograničenja.
- **TRUNCATE**: koristi se za brisanje podataka iz tablice, ali ne i same tablice ili njenih struktura.

MySQL vrste podataka

Vrsta podataka je način pohranjivanja podataka u bazu podataka.

CHAR i VARCHAR: CHAR i VARCHAR su vrste podataka za pohranu niza znakova. CHAR koristi fiksnu duljinu, dok VARCHAR koristi varijabilnu duljinu.

INT: INT (integer) je vrsta podataka za pohranu cijelih brojeva.

FLOAT i DOUBLE: FLOAT i DOUBLE su vrste podataka za pohranu decimalnih brojeva.

DATE, TIME i DATETIME: Ovo su vrste podataka koje se koriste za pohranu datuma i vremena.

TEXT i BLOB: TEXT i BLOB su vrste podataka koje se koriste za pohranu velikih količina teksta ili binarnih podataka.

BOOLEAN: BOOLEAN je vrsta podataka za pohranu logičkih vrijednosti true ili false.

Spajanja

INNER JOIN: Vraća samo one retke koji imaju podudaranja u oba tablice koje se spajaju.

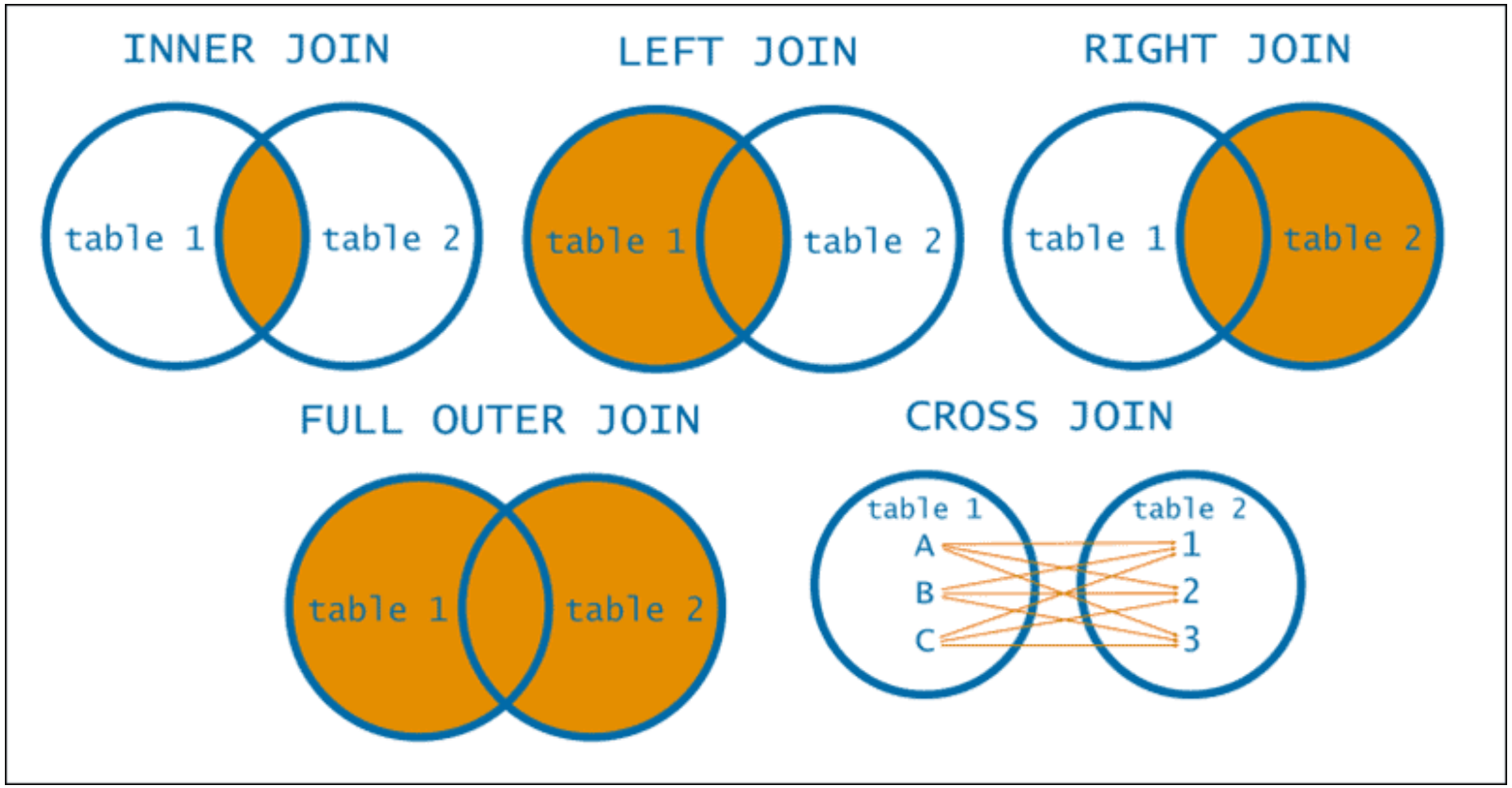
LEFT JOIN: Vraća sve retke iz lijeve tablice i pripadajuće podudarajuće retke iz desne tablice. Ako ne postoji podudaranje, za desnu tablicu se postavljaju NULL vrijednosti.

RIGHT JOIN: Vraća sve retke iz desne tablice i pripadajuće podudarajuće retke iz lijeve tablice. Ako ne postoji podudaranje, za lijevu tablicu se postavljaju NULL vrijednosti.

FULL OUTER JOIN: Vraća sve retke iz obje tablice i pripadajuće podudarajuće retke. Ako ne postoji podudaranje, za tablicu za koju ne postoji podudaranje postavljaju se NULL vrijednosti.

CROSS JOIN: Vraća sve moguće kombinacije između retka iz lijeve tablice i retka iz desne tablice.

Vizualni prikaz JOIN naredbe



Savjeti za rad s MySQL JOIN

- ✓Odaberite pravi tip JOIN-a za svoje potrebe.
- ✓MySQL nudi nekoliko vrsta JOIN-a, uključujući INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN itd. Ovisno o vašim potrebama, odaberite najprikladniji tip.
- ✓Nemojte preopteretiti svoj upit s previše JOIN-ova. Svako spajanje može povećati složenost upita i usporiti izvršavanje. Pazite da ne pretjerate sa spajanjima i da vam upit ostane čitljiv.
- ✓Provjerite svoje uvjete spajanja. Prije izvršavanja upita uvijek provjerite jesu li uvjeti spajanja točni. Ako nisu, to može dovesti do neispravnih rezultata.

Savjeti za rad s MySQL JOIN

- ✓ Upotrijebite indekse za poboljšanje performansi. Ako često koristite JOIN-ove u svojim upitima, vrijedi razmisliti o indeksiranju stupaca koji se koriste u uvjetima spajanja. To može značajno poboljšati performanse vaših upita.
- ✓ Pazite na NULL vrijednosti. Ako koristite LEFT JOIN ili RIGHT JOIN, provjerite postoji li mogućnost da se pojave NULL vrijednosti u rezultatima. Ako se to dogodi, budite spremni rukovati s njima u svojoj aplikaciji ili nadopuniti svoj upit uvjetima koji će ih isključiti.
- ✓ Testirajte svoje upite. Prije nego što ih stavite u produkciju, testirajte svoje upite na različitim skupovima podataka kako biste bili sigurni da dobivate točne i očekivane rezultate.

MySQL ugrađene funkcije

- MySQL ugrađene funkcije su funkcije koje su ugrađene u sam MySQL sustav i omogućuju različite operacije i manipulacije podacima.
- Neke od ugrađenih funkcija u MySQL-u su:

Agregatne funkcije — npr. COUNT(), SUM(), AVG(), MIN(), MAX() - koriste se za izračunavanje vrijednosti iz nekoliko redaka tablice.

Funkcije za datum i vrijeme — npr. NOW(), DATE(), TIME() - koriste se za manipulaciju datumima i vremenom u tablicama.

String funkcije — npr. CONCAT(), LENGTH(), LOWER(), UPPER(), SUBSTRING() - koriste se za manipulaciju tekstualnim podacima u tablicama.

Matematičke funkcije — npr. **ABS(), CEIL(), FLOOR(), RAND()** - koriste se za izvršavanje matematičkih operacija na podacima.

Logičke funkcije — npr. **IF(), IFNULL(), NULLIF()** - koriste se za izvođenje logičkih operacija na podacima.

Transakcije

- MySQL transakcije su skup radnji koje se izvode kao jedna logička cjelina, a to znači da se sve radnje izvršavaju ili ne izvršavaju u cijelosti.
- U slučaju da se dogodi bilo kakva pogreška tijekom transakcije, sve izmjene se vraćaju na stanje prije početka transakcije.

- **START TRANSACTION** označava početak transakcije
- **COMMIT** potvrđuje sve promjene koje su napravljene tijekom transakcije.
- **ROLLBACK** se koristi za poništavanje svih promjena koje su napravljene tijekom transakcije i vraćanje baze podataka u stanje prije početka transakcije.
- **SAVEPOINT** točke spremanja koriste se za označavanje točke u transakciji na koju se kasnije možete vratiti.

Primjer transakcije

```sql
START TRANSACTION;

INSERT INTO korisnici (ime, email, zaporka)
VALUES ('Ivan Ivić', 'ivan@example.com', 'password123');
UPDATE racuni SET stanje = stanje - 100 WHERE user_id = 1;
UPDATE racuni SET stanje = stanje + 100 WHERE user_id = 2;

COMMIT;
```

MySQL view

- MySQL View je virtualna tablica koja se sastoji od upita koji se izvode na jednoj ili više postojećih tablica u bazi podataka.
- View se kreira naredbom **CREATE VIEW**, a briše se naredbom **DROP VIEW**.
- View se mijenja naredbom **ALTER VIEW.**

- Preko pogleda se mogu mijenjati podaci.
- Pogledi se mogu spajati.
- Moguće je napraviti pogled od pogleda.

Indeksi

- Indeksi u MySQL-u su posebne strukture koje se koriste za poboljšanje performansi upita.
- Oni su slični indeksima u knjigama - omogućuju vam da brzo pronađete informacije bez pretraživanja cijele knjige.
- Indeksi se stvaraju na jednoj ili više kolona u tablici i pomoću njih se optimiziraju upiti koji uključuju te kolone u svojim uvjetima pretraživanja.
- Kada MySQL izvršava upit koji uključuje uvjete pretraživanja, on prvo provjerava indekse kako bi pronašao odgovarajuće redove u tablici.
- To može značajno smanjiti vrijeme izvođenja upita, posebno ako tablica ima veliki broj redova.

Indeksi su slični indeksima (kazalu) u knjigama - omogućuju brzo pronalaženje informacija bez pretraživanja cijele knjige.

Indeksi – sažetak

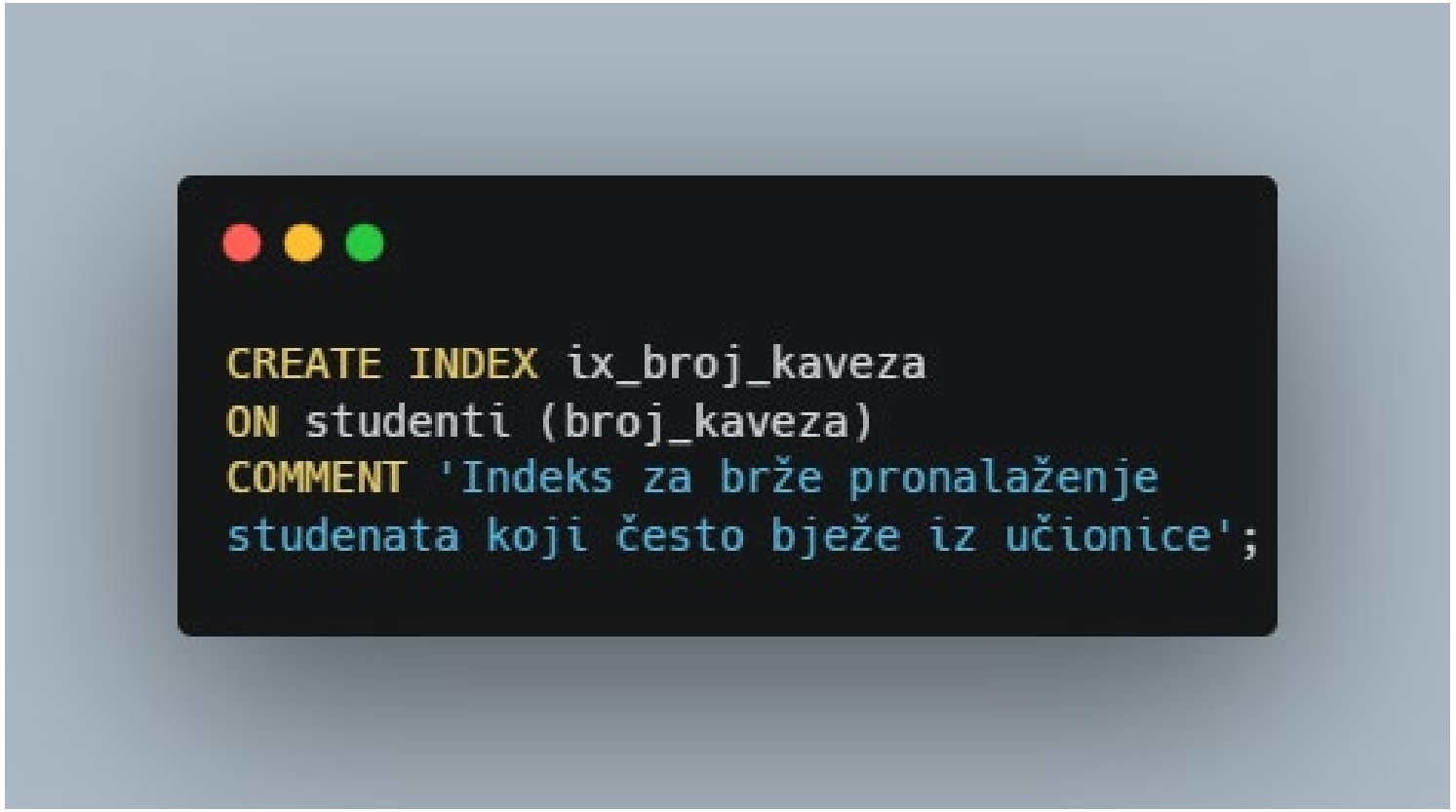
Indeksi se mogu koristiti za **ubrzavanje naredbi** SELECT, UPDATE i DELETE.

Naredba **CREATE INDEX** stvara indeks na jednom ili više stupaca tablice.

Indeksi se mogu obrisati pomoću naredbe **DROP INDEX**

Grupirani indeks (clustered index) određuje fizički redoslijed redaka u tablici.

Primjer!


CREATE INDEX ix_broj_kaveza
ON studenti (broj_kaveza)
COMMENT 'Indeks za brže pronalaženje
studenata koji često bježe iz učionice';

Okidač (*trigger*)

- Okidač je pohranjena procedura koja se automatski izvršava kada se dogodi događaj u bazi podataka.
- Okidač se izvršava kod sljedećih događaja.
  - Red je **umetnut** u tablicu.
  - Red je **izbrisan** iz tablice.
  - Redak je **ažuriran** u tablici.
- **OLD i NEW** koriste se za referenciranje vrijednosti stupaca koji se mijenjaju.
- Okidači se **ne mogu** koristiti za izmjenu podataka u istoj tablici u kojoj je okidač definiran.

Primjer okidača

```
CREATE TRIGGER azuriraj_kolicinu
AFTER INSERT ON narudzba
FOR EACH ROW
BEGIN
  UPDATE proizvod
  SET kolicina = kolicina - NEW.kolicina
  WHERE id = NEW.proizvod_id;
END;
```

Napredno PHP programiranje

Napredno PHP programiranje

- Napredno PHP programiranje obuhvaća različite koncepte i tehnike koje se koriste za razvoj skalabilnih, sigurnih i održivih aplikacija.
- Neke od tih koncepata i tehnika uključuju:
  - Objekti
  - Autoloader
  - Try-catch dohvaćanje grešaka
  - Obrasci dizajna singleton, factory, adapter i dr.
  - Povezivanje s bazom podataka, mysqli, PDO
  - Testiranje
  - Paketni menadžer Composer
  - MVC arhitektura

Objektno orijentirano programiranje (OOP)

- OOP je pristup programiranju koji se temelji na objektima i njihovim međusobnim interakcijama.
- U PHP-u se OOP koncepti koriste za organiziranje koda u višestruko iskoristive module i za smanjenje ponavljanja koda.
- Objekti su instance klasa u PHP-u.
- Klase definiraju svojstva (varijable) i metode (funkcije), a objekti su njihove konkretne implementacije.
- OOP koristi koncepte *apstrakcije, enkapsulacije, nasljeđivanja i polimorfizma.*

Primjer PHP objektnog koda

```php
class Osoba {
  protected $ime;
  protected $prezime;

  public function __construct($ime, $prezime) {
    $this->ime = $ime;
    $this->prezime = $prezime;
  }

  public function predstavi_se() {
    echo "Ja sam " . $this->ime . " " . $this->prezime . ".";
  }
}

class Polaznik extends Osoba {
  public function polozi_ispit() {
    $uspjeh = rand(0, 1); // generiraj nasumičan broj 0 ili 1
    if ($uspjeh) {
      echo "Čestitam, uspješno ste položili ispit! :)";
    } else {
      echo "Nažalost, niste uspjeli položiti ispit. :( Pokušajte ponovno!";
    }
  }
}

$polaznik = new Polaznik("Marko", "Markić");
$polaznik->predstavi_se(); // ispisuje "Ja sam Marko Markić."
$polaznik->polozi_ispit(); // ispisuje poruku o uspješnom ili neuspješnom položenju ispita
```

Zadatak

Napravite klasu „**HrvatskeZeljeznice**” koja ima metodu **vozi()** i svojstvo **kasnjenje**.

Kreirajte objekt tipa HrvatskeZeljeznice

```php
<?php

class HrvatskeZeljeznice {
  private $naziv = "Hrvatske željeznice";
  private $kasnjenje = true;

  function getNaziv() {
    return $this->naziv;
  }

  function vozi() {
    if ($this->kasnjenje) {
      return "Došli smo na odredište sa samo 2 sata kašnjenja, hvala što ste putovati s " . $this->naziv;
    } else {
      return "Došli smo na vrijeme, ali to nije u duhu " . $this->naziv . " pa ćemo se malo zadržati na stanici";
    }
  }
}

$putovanje = new HrvatskeZeljeznice();
echo $putovanje->vozi();
```

Objekti - sažetak

- U programiranju, objekti su instance klasa koje mogu imati svojstva i metode.
- Kreira se korištenjem ključne riječi **"new"** nakon koje slijedi naziv klase.
- Svojstvima objekta u PHP-u pristupate korištenjem operatora "->" iza kojeg slijedi naziv svojstva.
- Enkapsulacija je način skrivanja detalja implementacije klase od vanjskog svijeta.
- Konstruktor je metoda koja se poziva kada se kreira objekt.

Enkapsulacija primjer

Idete na odmor?

Provedite pola odmora u našem vlaku.

```php
<?php

class HrvatskeZeljeznice {
  private $kasnjenje = true;

  public function UobicajeniDolazak() {
    // Ova metoda vraća vrijeme dolaska vlaka, maskirajući kašnjenje
    // Hrvatskih željeznica kao "uobičajeni dolazak".
    $vrijemeDolaska = new DateTime();

    if ($this->kasnjenje) {
      $vrijemeDolaska->modify('+2 hours');
      return $vrijemeDolaska->format('H:i') . " (uobičajeni dolazak)";
    } else {
      return $vrijemeDolaska->format('H:i');
    }
  }
}

// Kreiramo objekt HrvatskeZeljeznice
$putovanje = new HrvatskeZeljeznice();

// Prikazujemo vrijeme dolaska vlaka, maskirajući kašnjenje kao "uobičajeni dolazak"
echo "Vrijeme dolaska: " . $putovanje->UobicajeniDolazak() . "\n";
```

Objekti - sažetak

- Apstraktna klasa je nacrt za ostale klase koje ju nasljeđuju.
- Sučelje (interface) je popis metoda koje klasa mora implementirati.
- Klasa može implementirati više sučelja.
- Imenski prostor može biti samo jedan po datoteci
- Imenski prostor služi poboljšanju performansi PHP aplikacija.
- Nasljeđivanje omogućuje stvaranje novih klasa na temelju postojećih, gdje nova klasa nasljeđuje svojstva i metode roditeljske klase.

Apstrakcija primjer

```php
<?php

// Definiramo apstraktnu klasu prijevoznog sredstva
abstract class PrijevoznoSredstvo {
  abstract public function vrijemeDolaska();
}

// Definiramo konkretnu podklasu Hrvatske željeznice, koja nasljeđuje apstraktnu klasu PrijevoznoSredstvo
class HrvatskeZeljeznice extends PrijevoznoSredstvo {
  public function vrijemeDolaska() {
    return date('H:i', strtotime('+2 hours')) . " (uobičajeni dolazak)";
  }
}

// Kreiramo objekt HrvatskeZeljeznice i prikazujemo vrijeme dolaska vlaka koristeći apstrakciju
$putovanje = new HrvatskeZeljeznice();
echo "Vrijeme dolaska vlaka Hrvatskih željeznica: " . $putovanje->vrijemeDolaska() . "\n";
```

PHP automatski učitavač (*autoloader*)

- PHP autoloader je funkcija koja automatski učitava klase i datoteke kad se pozovu u kodu.
- Svrha autoloadera je olakšati razvoj i održavanje aplikacije jer programer ne mora ručno uključivati sve klase i datoteke koje su mu potrebne.
- Smanjuje količinu koda koji trebate napisati.
- Vaš kod čini učinkovitijim.
- Čini vaš kod lakšim za održavanje.

Primjer autoloader

```php
// definiranje autoloader funkcije
function my_autoloader($class_name) {
    include 'classes/' . $class_name . '.php';
}

// registriranje autoloadera
spl_autoload_register('my_autoloader');

// kreiranje objekta Student
$student = new Student('Marko', 'Marković', '12345');

// pozivanje metode na objektu
$student->poloziIspit('Programiranje');
```

Try-catch

- Try-catch blok u PHP-u omogućuje rukovanje iznimkama koje se javljaju tijekom izvođenja koda.
- Try blok sadrži kod koji treba biti izvršen, a catch blok hvata iznimke koje su izbačene tijekom izvođenja koda unutar try bloka.
  - Omogućuje programeru da se bavi iznimnim situacijama na prikladan način, umjesto da program jednostavno prestane raditi ako se dogodi neočekivana pogreška.

Try-catch primjer

```php
class Student {
  public $ime;
  public $ocjena;

  public function polaganjeIspita($ocjena) {
    if ($ocjena < 5) {
      throw new Exception("Student nije položio ispit!");
    } else {
      $this->ocjena = $ocjena;
      echo "Čestitamo, student " . $this->ime
        . " je položio ispit sa ocjenom "
        . $ocjena . "!";
    }
  }
}

$student = new Student();
$student->ime = "Pero";



try {
  $student->polaganjeIspita(3);
} catch (Exception $e) {
  echo "Došlo je do greške: " . $e->getMessage();
}
```

Singleton predložak

- PHP Singleton predložak (engl. *Singleton pattern*) je obrazac koji osigurava da postoji samo jedna instanca određene klase u cijeloj aplikaciji.
- U PHP-u, Singleton obrazac se često koristi za stvaranje objekata koji su potrebni za pristup bazi podataka ili za konfiguracijske postavke aplikacije.
- Singleton obrazac se sastoji od jedne klase koja ima statičku metodu za vraćanje jedine instance klase.
- Ova metoda stvara novu instancu klase samo ako još nije postojala, a inače vraća postojeću instancu.

Singleton predložak

```php
class MackaSingleton {
   private static $instanca;
   private $brojNapada;

   private function __construct() {
      $this->brojNapada = 0;
   }

   public static function dohvatiInstancu() {
      if (!isset(self::$instanca)) {
         self::$instanca = new MackaSingleton();
      }
      return self::$instanca;
   }

   public function napadniMisa() {
      $this->brojNapada++;
      echo "Mačka napada miša! Broj napada: "
        . $this->brojNapada . "\n";
   }
}

// Korištenje Singletona
$maca = MackaSingleton::dohvatiInstancu();
$maca->napadniMisa();

$maca2 = MackaSingleton::dohvatiInstancu();
$maca2->napadniMisa();

// Ispis:
// Mačka napada miša! Broj napada: 1
// Mačka napada miša! Broj napada: 2
```

Factory predložak

- Factory pattern u PHP-u je dizajnerski obrazac koji se koristi za kreiranje objekata bez eksplicitnog navođenja njihove klase.
- Kada koristiti factory pattern:
  - Kada imate više implementacija istog sučelja, a ne želite da korisnik mora izravno instancirati objekte
  - Kada želite da se klijentski kod bavi samo sučeljem, a ne s pojedinim implementacijama

Primjer PHP factory predloška

za kreiranje različitih vrsta životinja poput psa, mačke i miša

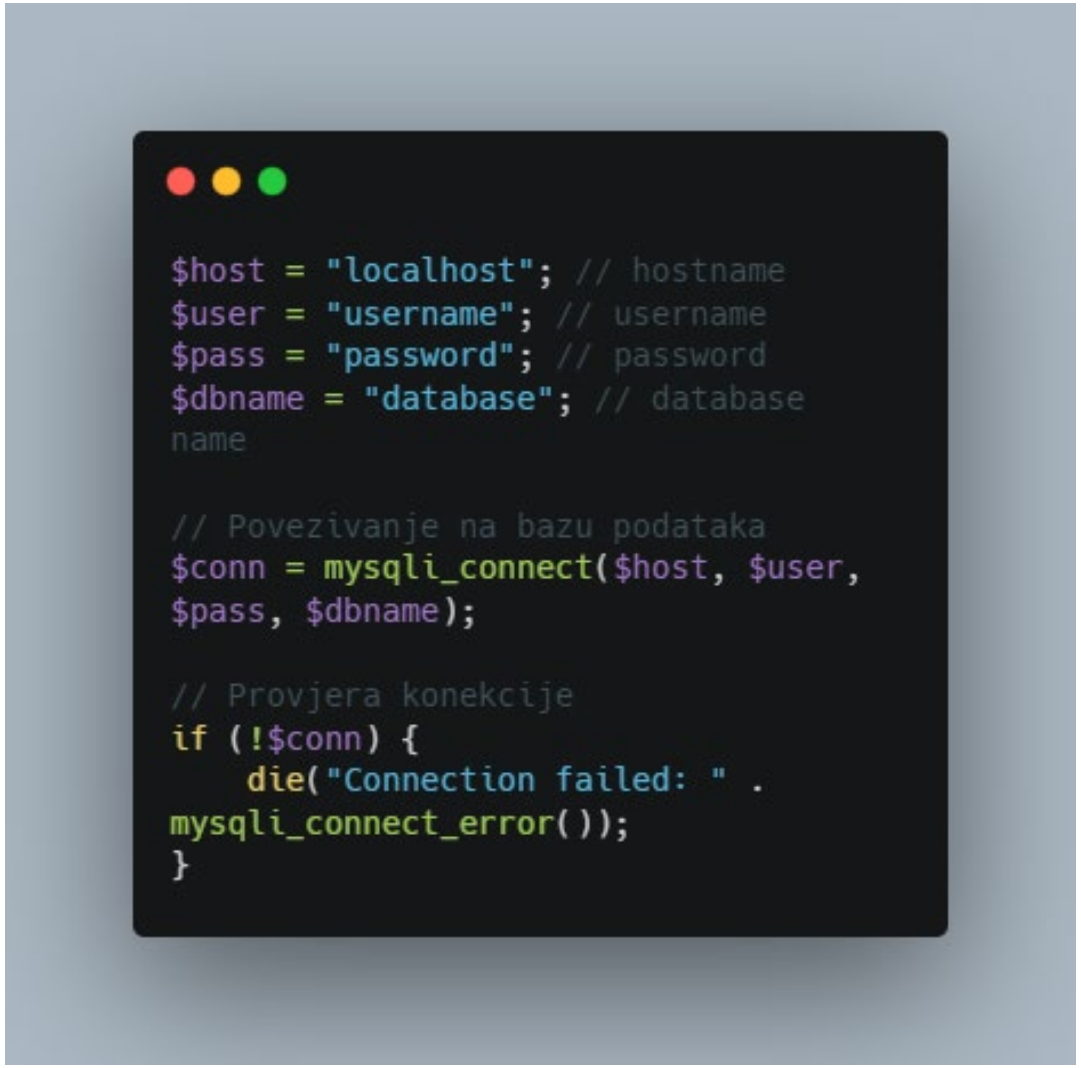
```php
// Kreiramo sučelje Animal koje će
// implementirati sve životinje
interface Animal {
    public function makeSound();
}

// Kreiramo klase za svaku vrstu
// životinje koju želimo podržati
class Dog implements Animal {
    public function makeSound() {
        echo "Vau vau!";
    }
}

class Cat implements Animal {
    public function makeSound() {
        echo "Mijau mijau!";
    }
}

class Mouse implements Animal {
    public function makeSound() {
        echo "Cvrc cvrc!";
    }
}
```

```php
// Kreiramo Factory klasu koja će
stvoriti životinje na temelju
proslijeđenog tipa
class AnimalFactory {
    public static function
createAnimal($type) {
        switch ($type) {
            case 'dog':
                return new Dog();
                break;
            case 'cat':
                return new Cat();
                break;
            case 'mouse':
                return new Mouse();
                break;
            default:
                throw new
Exception("Unsupported animal
type.");
        }
    }
}

// Korištenje factory predloška za
stvaranje objekata životinja
$dog =
AnimalFactory::createAnimal('dog');
$cat =
AnimalFactory::createAnimal('cat');
$mouse =
AnimalFactory::createAnimal('mouse');

// Poziv funkcije makeSound() za
svaku životinju
$dog->makeSound(); // Ispisuje "Vau
vau!"
$cat->makeSound(); // Ispisuje "Mijau
mijau!"
$mouse->makeSound(); // Ispisuje
"Cvrc cvrc!"
```

MySQLi

- **MySQLi** *(MySQL Improved Extension)* je proširenje PHP-a koje omogućuje komunikaciju sa MySQL bazama podataka.
- Za povezivanje na MySQL bazu podataka koristi se funkcija mysqli_connect() koja prima četiri parametra: **hostname, username, password i database**.
- Nakon uspješne konekcije na bazu podataka, možemo izvršavati upite na bazi podataka koristeći funkcije poput **mysqli_query(), mysqli_fetch_assoc(), mysqli_fetch_array()** i druge.
- **mysqli_connect_error()** je funkcija koja se koristi za provjeru je li MySQLi veza bila uspješna

Primjer koda za uspostavu veze s MySQL bazom podataka



```php
$host = "localhost"; // hostname
$user = "username"; // username
$pass = "password"; // password
$dbname = "database"; // database
name

// Povezivanje na bazu podataka
$conn = mysqli_connect($host, $user,
$pass, $dbname);

// Provjera konekcije
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
```

PDO (*PHP Data Objects*)

- PDO je PHP ekstenzija koja omogućuje programerima da se povežu na različite vrste baza podataka, uključujući MySQL, PostgreSQL, Oracle i druge.
- PDO nudi konzistentno sučelje za rad sa različitim bazama podataka, što znači da se isti kod može koristiti za rad sa različitim bazama podataka, bez potrebe za promjenom koda.
- Jedna od glavnih prednosti korištenja PDO-a je da se može koristiti prepared statement mehanizam za izbjegavanje SQL injection napada.
- **prepare()** je metoda koja se koristi za pripremu SQL upita-
- **execute()** je metoda koja se koristi za izvršavanje pripremljenog PDO upita.
- PDO veze se automatski zatvaraju kada skripta završi.

Primjer spajanja na bazu podataka uz pomoć PDO

```php
$host = 'localhost';
$dbname = 'studenti';
$charset = 'utf8mb4';
$username = 'root';
$password = 'mypassword';

$dsn = "mysql:host=$host;dbname=$dbname;charset=$charset";

try {
    $pdo = new PDO($dsn, $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo "Neuspješno spajanje na bazu: " . $e->getMessage();
    exit();
}
```

Testiranje

- TDD (*Test Driven Development*) je pristup razvoju softvera koji se fokusira na pisanje testova prije pisanja samog koda.
- Cilj je osigurati da se softver ispravno ponaša u svim mogućim scenarijima, prije nego što se uopće krene s implementacijom.
- Testiranje softvera postaje sve važnije kako se zahtjevi za softverom povećavaju, a kvaliteta i pouzdanost postaju ključni faktori u uspjehu softverskih projekata. Stoga, ako želite biti uspješan programer, vrijedi uložiti vrijeme i napor u učenje PHPUnit-a i jediničnog testiranja u PHP-u.

1

Prvo, napišimo test za funkcionalnost koja bi trebala omogućiti mački da uhvati miša:

```
test('Mačka uspješno lovi miša', function() {
  $mouse = new Mouse();
  $cat = new Cat();
  $cat->hunt($mouse);
  expect($mouse->isCaught())->toBeTrue();
});
```

2

Zatim, implementirajmo funkciju "hunt" u klasi "Cat":

```
class Cat {
  public function hunt(Mouse $mouse) {
    $mouse->catch();
  }
}
```

3

Nakon toga, pokrenemo test i vidimo da ‘ne prolazi’ jer klasa "Mouse" nema implementiranu funkciju "catch". Dakle, dodajmo funkciju "catch()" u klasu "Mouse":

```
class Mouse {
  private $isCaught = false;

  public function catch() {
    $this->isCaught = true;
  }

  public function isCaught() {
    return $this->isCaught;
  }
}
```

Sada kada pokrenemo test, vidimo da prolazi i da mačka uspješno lovi miša!

Zadatak

Napišite test koji uspoređuje stvarni dolazak vlaka s očekivanim.

```php
<?php

use PHPUnit\Framework\TestCase;

class HrvatskeZeljezniceTest extends TestCase {
  public function testVrijemeDolaskaKasnjenje() {

    // Stvaramo objekt Hrvatske željeznice koji kasni
    $putovanje = new HrvatskeZeljeznice(true);

    // Očekujemo dolazak vlaka tri sata kasnije
    $ocekivanoVrijeme = date('H:i', strtotime('+3 hours'));

    $this->assertEquals($ocekivanoVrijeme, $putovanje->vrijemeDolaska(),
                        "Iznenađenje bi bilo da je došao na vrijeme!");
  }
}
```

PHPUnit

- PHPUnit je popularni okvir za **jedinično testiranje** u PHP-u.
- Koristi se za testiranje pojedinačnih jedinica koda, kao što su funkcije ili metode, kako bi se osigurala ispravnost njihove implementacije.

PHPUnit često korištene metode

assertTrue(): Provjerava da je uvjet istinit (true).

assertFalse(): Provjerava da je uvjet lažan (false).

assertEquals (): Provjerava da su očekivana i stvarna vrijednost jednake.

assertSame (): Provjerava da su očekivana i stvarna vrijednost istog tipa i vrijednosti.

PHPUnit često korištene metode

assertGreaterThan(): Provjerava da je stvarna vrijednost veća od očekivane vrijednosti.

assertLessThan(): Provjerava da je stvarna vrijednost manja od očekivane vrijednosti.

assertArrayHasKey(): Provjerava da ključ postoji u polju.

assertContains(): Provjerava da se vrijednost nalazi u nizu.

assertEmpty(): Provjerava da je vrijednost prazna.

assertNotEmpty(): Provjerava da vrijednost nije prazna.

Što je to MVC?

- **MVC (Model-View-Controller)** je arhitektonski obrazac koji se koristi za organiziranje i strukturiranje PHP web aplikacija.
- Konceptualno, MVC razdvaja aplikaciju na tri glavna dijela: **Model, View i Controller.**

Model: Model je sloj koji se bavi poslovnom logikom aplikacije i podacima. Ovdje se obično nalaze klase koje predstavljaju entitete i **pristupaju bazi podataka**.

Controller: Controller je sloj koji upravlja komunikacijom između Modela i Viewa. Ovdje se obično nalaze PHP skripte koje obrađuju ulazne podatke, a zatim koriste Model kako bi dohvatili, ažurirali ili obrisali podatke. Nakon toga, Controller generira odgovarajući View koji korisniku prikazuje informacije

View: View se koristi za prikazivanje informacija korisnicima. Obično se sastoji od HTML-a i CSS-a. Ovdje se također nalaze predlošci (templates) koji predstavljaju strukture prikaza.

composer.json VS composer.lock

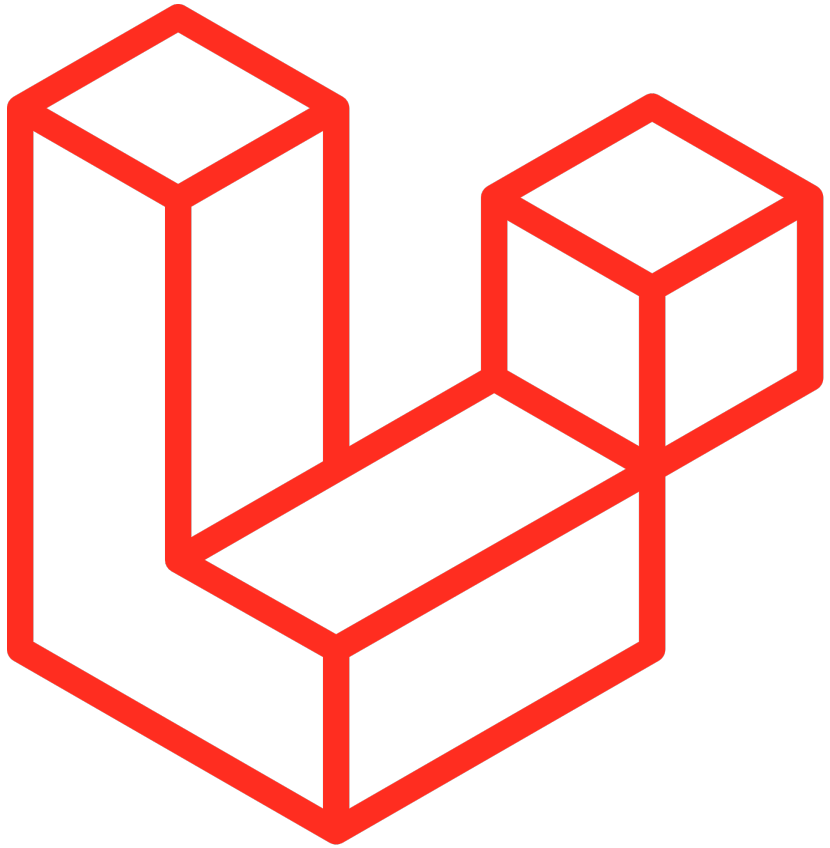
"**composer.json**" je datoteka koja opisuje ovisnosti vašeg projekta. U njoj definirate pakete i njihove verzije koje vaš projekt zahtijeva za rad. Također se mogu navesti i drugi parametri vezani uz instalaciju paketa (npr. putanje do datoteka, skripte koje se izvršavaju nakon instalacije, itd.).

"**composer.lock**" je generirana datoteka koja sadrži točne verzije svih paketa koji su trenutno instalirani u vašem projektu. Ova datoteka se koristi kako bi se osiguralo da se iste verzije paketa instaliraju i na drugim računalima ili serverima.

Radni okvir Laravel

Laravel

- PHP Laravel je popularni *open-source* web okvir za razvoj aplikacija napisanih u programskom jeziku PHP.
- https://laravel.com/
- https://github.com/laravel/framework
- Konfiguracijska datoteka je **.env**
- Nativno podržava PostgreSQL,SQLite i MySQL baze



Izvor: Google Images

Prednosti Laravela uključuju

- **Snažan sustav za rutiranje** - Laravel omogućuje jednostavno upravljanje rutama, što olakšava izgradnju aplikacija s različitim rutama.
- **Ugrađena autentikacija** - Laravel dolazi s ugrađenim sustavom autentikacije koji programerima omogućuje jednostavno upravljanje pristupom korisnika i upravljanjem sesijama.
- **Eloquent ORM** - Laravel dolazi s Eloquent ORM-om, koji omogućuje programerima rad s bazama podataka koristeći PHP sintaksu, što olakšava upravljanje i manipuliranje podacima u bazi podataka.
- **Blade templating engine** - Laravel dolazi s moćnim Blade templating engine-om koji omogućuje programerima jednostavno kreiranje sjajnih i dinamičkih predložaka za svoje aplikacije.
- **Velika zajednica** - Laravel ima veliku zajednicu programera koji doprinose razvoju okvira i nude podršku drugim programerima.

Kako započeti novi Laravel projekt

Instalirajte PHP i Composer: Laravel zahtijeva PHP verziju 7.4 ili noviju, pa ako ga nemate instaliranog, to prvo trebate učiniti. Također, potrebno je instalirati Composer, koji je alat za upravljanje ovisnostima u PHP projektima.

Instalirajte Laravel putem Composer-a: Nakon što imate PHP i Composer instalirane, sljedeći korak je instaliranje Laravela putem Composer-a. Možete to učiniti pokretanjem naredbe "composer create-project --prefer-dist laravel/laravel ime-projekta" u terminalu, gdje "ime-projekta" predstavlja naziv vašeg novog projekta.

Konfigurirajte bazu podataka: Laravel dolazi s predloškom datoteke .env koja sadrži postavke baze podataka. Otvorite ovu datoteku i konfigurirajte svoju bazu podataka prema svojim potrebama.

Sljedeći korak

Pokrenite migracije: Laravel koristi migracije za upravljanje shemom baze podataka. Ako planirate koristiti bazu podataka u vašem projektu, potrebno je pokrenuti migracije. To možete učiniti pokretanjem naredbe "php artisan migrate" u terminalu.

Kreirajte modele, kontrolere i rute: Nakon što ste instalirali Laravel i konfigurirali bazu podataka, sljedeći korak je kreiranje modela, kontrolera i ruta. Modeli predstavljaju tablice u bazi podataka, kontroleri obrađuju zahtjeve korisnika, a rute omogućuju usmjeravanje zahtjeva na odgovarajuće kontrolore.

Najčešće korištene naredbe

php artisan serve: Ova naredba pokreće ugrađeni web server i omogućuje vam da pregledate vašu Laravel aplikaciju u pregledniku. Server će se pokrenuti na adresi http://localhost:8000/ prema zadanim postavkama.

php artisan make:model NazivModela: Ova naredba stvara novi model unutar app/Models direktorija vašeg projekta. Možete dodati opciju --migration kako biste stvorili i migraciju za model.

php artisan make:controller NazivKontrolera: Ova naredba stvara novi kontroler unutar app/Http/Controllers direktorija vašeg projekta.

php artisan make:migration naziv_migracije: Ova naredba stvara novu migraciju u direktoriju database/migrations vašeg projekta. Migracije se koriste za promjene sheme baze podataka.

Najčešće korištene naredbe

php artisan migrate: Ova naredba pokreće sve migracije koje još nisu pokrenute na vašoj bazi podataka.

php artisan tinker: Ova naredba pokreće Tinker, interaktivno sučelje za rad s vašom aplikacijom. To vam omogućuje da testirate razne naredbe i funkcionalnosti vašeg koda.

php artisan route:list: Ova naredba prikazuje popis svih ruta koje ste definirali u vašoj aplikaciji.

php artisan make:middleware NazivMiddlewarea: Ova naredba stvara novi middleware u app/Http/Middleware direktoriju vašeg projekta. Middleware se koristi za dodavanje funkcionalnosti na zahtjev i odgovor na putu do odredišta.

Middleware

- Middleware u Laravelu je izvršni sloj koji se nalazi između zahtjeva i odgovora.
  - Kada klijent šalje zahtjev, on prvo prolazi kroz *middleware* koji ga obrađuje, a nakon toga se generira odgovor.
- Middleware se koristi za filtriranje, autorizaciju, manipulaciju zahtjeva i/ili odgovora te za dodavanje raznih funkcionalnosti i slojeva sigurnosti.
- Middleware se može dodijeliti pojedinačnim rutama ili grupama ruta, a može se također primijeniti globalno na sve rute u aplikaciji.
  - Kada se middleware dodijeli ruti, on se izvršava samo kada se ta ruta pozove.

Često korišteni middleware

Authenticate - koristi se za provjeru autentičnosti korisnika i sprečavanje pristupa neautoriziranim korisnicima. **Metoda je Auth::login()**

ThrottleRequests - koristi se za ograničavanje broja zahtjeva po vremenskom periodu, kako bi se spriječio DDoS napad ili preopterećenje servera

VerifyCsrfToken - koristi se za provjeru CSRF tokena kako bi se zaštitilo korisničko sučelje od CSRF napada

EncryptCookies - koristi se za enkripciju kolačića kako bi se zaštitili korisnički podaci

Cors - koristi se za omogućavanje Cross-Origin Resource Sharing-a (CORS) i sprečavanje blokiranja CORS-a u pregledniku

Cache - koristi se za spremanje izračunatih vrijednosti u predmemoriju kako bi se smanjilo vrijeme odziva aplikacije

Rute

- Rute u Laravelu su mehanizam za mapiranje URL adresa na odgovarajuće kontrolere i metode.
- **redirect()** je metoda za preusmjeravanje korisnika na drugi URL u Laravel kontroleru.
- **{param}:** metoda koristi za definiranje parametra rute
- **Metode za grupiranje ruta**: **Route::prefix(), Route::middleware(), Route::group()**
- U Laravelu, rute se definiraju u datoteci routes/web.php za rute web aplikacije i u datoteci routes/api.php za API rute.
- Laravel također podržava grupiranje ruta po određenom prefiksu, što je korisno kada želite imati isti prefiks za sve rute unutar određene skupine.

Primjer GET rute

- Ova ruta će omogućiti pristup metodi index u AlgebraController-u preko URL adrese /api/algebra, a imenovana je algebra.index. Varijabla studentID će biti poslana s vrijednosti 1234.
- Također, middleware auth osigurava da samo autentificirani korisnici mogu pristupiti ovoj ruti.

```php
Route::prefix('api')
  ->middleware('auth')->name('algebra.')
  ->group(function () {
    Route::get('/algebra', 'AlgebraController@index')
      ->with('studentID', 1234)
      ->name('index');
});
```

Migracije

- Migracija je PHP datoteka koja sadrži upute za promjene u bazi podataka, kao što su stvaranje novih tablica, dodavanje novih stupaca, indeksa ili izmjene postojećih tablica.
- Primjer Laravel migracije za stvaranje users tablice sa id, name, email i password stupcima:
- Ova migracija se može izvršiti pokretanjem **php artisan migrate** naredbe. Ako je potrebno, migracije se također mogu poništiti pomoću naredbe **php artisan migrate:rollback**. Migracije se također mogu stvoriti pomoću naredbe **php artisan make:migration**.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint
$table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

```php
use Illuminate\Database\Seeder;
use App\Models\User;

class UserSeeder extends Seeder
{
    public function run()
    {
        User::insert([
            ['name' => 'Marinko',
             'email' => 'marinko@example.com',
             'password' => bcrypt('password')],

            ['name' => 'Janica',
             'email' => 'janica@example.com',
             'password' => bcrypt('password')],
        ]);
    }
}
```

Seederi

- Laravel Seeder je mehanizam koji vam omogućuje da popunite bazu podataka s početnim podacima.
- Za pokretanje određenog seeder-a koristimo **php artisan db:seed --class=SeederName**

Laravel Model

- U Laravelu, Model je konvencija dizajna softvera koja predstavlja tablicu u bazi podataka. Modeli se koriste za izvršavanje operacija na bazi podataka, kao što su čitanje, stvaranje, ažuriranje i brisanje podataka.
- U ovom primjeru, uz svojstvo $fillable samo polja name, email i password mogu se popuniti masovno. Sva ostala polja, kao što su ID i vrijeme stvaranja, bit će ignorirana.

```php
use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    protected $fillable = ['name', 'email', 'password'];
}
```

Relacije modela

- U Laravelu, relacije se koriste za povezivanje različitih modela i omogućavanje izvođenja složenih upita koji uključuju podatke iz više modela.

- *Belongs To*: Belongs To relacija koristi se kada jedan model pripada drugom modelu.
- *Has One*: Has One relacija koristi se kada jedan model ima samo jedan povezani model.
- *Has Many*: Has Many relacija koristi se kada jedan model ima više povezanih modela.
- *Many To Many*: Many To Many relacija koristi se kada jedan model može imati više povezanih modela i obrnuto.
- *Has Many Through*: Has Many Through relacija koristi se kada imamo tri modela i želimo dohvatiti podatke iz trećeg modela preko prvog i drugog modela.

Belongs To

koristi se kada jedan model pripada drugom modelu.

Has One

koristi se kada jedan model ima samo jedan povezani model.

Has Many

koristi se kada jedan model ima više povezanih modela.

```php
public function posts()
{
    return $this->hasMany('App\Post');
}
```

Many To Many

relacija koristi se kada jedan model može imati više povezanih modela i obrnuto.

Has Many Through

koristi se kada imamo tri modela i želimo dohvatiti podatke iz trećeg modela preko prvog i drugog modela.

```php
public function posts()
{
    return $this->hasManyThrough('App\Post', 'App\User');
}
```

Eloquent

- Eloquent je ORM (*Object-Relational Mapping*) ugrađen u radni okvir Laravel koji omogućava programerima da rade s bazama podataka **koristeći objekte i metode** umjesto SQL upita.

```php
//Dohvati sve studente iz baze podataka:
$students = Student::all();

//Dohvati studente s određenim imenom:
$students = Student::where('name', 'Željko')->get();

//Dohvati studente sortirane prema ocjenama:
$students = Student::orderBy('ocjena', 'desc')->get();

// Dohvati studenta s najvišom ocjenom:
$student = Student::orderBy('grade', 'desc')->first();

//Dohvati prvih pet studenata sortiranih prema ocjenama:
$students = Student::orderBy('ocjena', 'desc')->take(5)->get();

//Dohvati studente s ocjenama između 3 i 5:
$students = Student::whereBetween('ocjena', [3, 5])->get();

//Dohvati studente koji su upisani na određeni smjer:
$students = Student::whereHas('smjer', function($query) {
    $query->where('name', 'PHP BackEnd Dev');
})->get();

//Dohvati studente koji su upisani na određeni smjer i sortirani prema imenima:
$students = Student::whereHas('course', function($query) {
    $query->where('name', 'PHP BackEnd Dev'');
})->orderBy('name')->get();
```

Klasa DB

- Klasa DB u Laravelu predstavlja centralni pristupni točku za interakciju s bazom podataka.
- Ova klasa omogućuje izvršavanje SQL upita nad bazom podataka bez potrebe za korištenjem modela ili Eloquent ORM-a

```php
//Selekcija svih studenata:
$students = DB::table('student')->get();

//Selekcija samo imena i prezimena svih studenata:
$students = DB::table('student')->select('ime', 'prezime')->get();

//Selekcija studenata koji pohađaju smjer "Programiranje" i sortir
po imenu u uzlaznom redoslijedu:
$students = DB::table('student')
                ->where('smjer', 'Programiranje')
                ->orderBy('ime', 'asc')
                ->get();

//Selekcija studenata koji su upisani nakon 2020. godine i sortira
po datumu upisa u silaznom redoslijedu:
$students = DB::table('student')
                ->whereYear('datum_upisa', '>', 2020)
                ->orderBy('datum_upisa', 'desc')
                ->get();

//Selekcija studenata koji pohađaju smjer "Engleski jezik" i imaju
prebivalište u Zagrebu:
$students = DB::table('student')
                ->where('smjer', 'Engleski jezik')
                ->where('prebivaliste', 'Zagreb')
                ->get();
```

Laravel testiranje

- Laravel testiranje se odnosi na proces testiranja Laravel aplikacija kako bi se osigurala njihova funkcionalnost, izdržljivost i skalabilnost.
- Laravel testiranje omogućuje programerima da automatiziraju testiranje aplikacija i osiguraju da aplikacija radi ispravno prije nego što se uvede u produkcijsko okruženje.
- Laravel testiranje uključuje testiranje različitih dijelova aplikacije, uključujući rute, kontrolere, modele, migracije, seedere i druge dijelove aplikacije. Testovi se mogu pisati pomoću različitih tehnika testiranja, kao što su **jedinično testiranje, funkcionalno testiranje i integracijsko testiranje**.
- Laravel testiranje se može izvršiti pomoću naredbe "**php artisan test**" koja će pokrenuti sve testove u direktoriju "tests". Također je moguće pokrenuti pojedinačne testove pomoću naredbe "**php artisan test --filter <test-name>**".

Laravel Dusk

- Laravel Dusk je paket za testiranje koji se koristi za automatizirano testiranje web aplikacija.
- Dusk koristi Selenium WebDriver za emuliranje interakcije korisnika sa web stranicom i provjeru radi li aplikacija ispravno.

```php
//visit($url) - Ova funkcija otvara web stranicu sa zadanim URL-om.
$this->browse(function ($browser) {
    $browser->visit('/login');
});

//type($selector, $value) - Ova funkcija simulira unošenje teksta u polje na web stranici. Argument $selector predstavlja CSS selektor elementa, a $value je tekst koji želimo unijeti.
$this->browse(function ($browser) {
    $browser->visit('/login')
            ->type('input[name="email"]', 'john@example.com')
            ->type('input[name="password"]', 'password123')
            ->press('Login')
            ->assertPathIs('/dashboard');
});
```

Primjer Dusk naredbi

- **visit($url):** otvara stranicu na zadanom URL-u
- **assertSee($text):** provjerava da li se zadani tekst prikazuje na stranici
- **type($selector, $text):** upisuje zadani tekst u formu ili polje označeno određenim selektorom
- **press($button):** simulira klik na gumb sa zadanim tekstom ili selektorom
- **seePageIs($url):** provjerava da li se trenutno prikazuje stranica sa zadanim URL-om
- **assertTitle($title):** provjerava da li je naslov trenutne stranice jednak zadanoj vrijednosti
- **clickLink($linkText):** simulira klik na link sa zadanim tekstom
- **screenshot($fileName):** sprema screenshot trenutne stranice u zadani file

```php
//click($selector) - Ova funkcija simulira klik na element na
web stranici.
$this->browse(function ($browser) {
    $browser->visit('/dashboard')
            ->click('.logout')
            ->assertPathIs('/login');
});

//assertSee($text) - Ova funkcija provjerava da li se neki
tekst nalazi na web stranici.
$this->browse(function ($browser) {
    $browser->visit('/dashboard')
            ->assertSee('Welcome to the dashboard!');
});

//screenshot($filename) - Ova funkcija sprema snimak trenutnog
stanja web stranice kao sliku. Argument $filename predstavlja
naziv datoteke u koju će slika biti spremljena.
$this->browse(function ($browser) {
    $browser->visit('/dashboard')
            ->screenshot('dashboard.png');
});
```

Prijenos podataka na produkcijsko okruženje

Prijenos podataka na produkcijsko okruženje

- Implementacija u kontekstu PHP *backend developmenta* se odnosi na proces u kojem se kod koji je razvijen u fazi razvoja prenosi i postavlja na produkcijsko okruženje.
- To znači da se kod postavlja na web server kako bi aplikacija mogla biti dostupna korisnicima.

Proces implementacije uključuje sljedeće korake:

Postavljanje produkcijskog okruženja: potrebno je postaviti server na kojem će se aplikacija izvoditi i provjeriti da li su sve potrebne konfiguracije ispunjene (npr. PHP verzija, potrebne PHP ekstenzije, konfiguracija baze podataka itd.).

Postavljanje koda na server: kod se kopira na server, najčešće putem FTP-a ili nekog drugog sustava za upravljanje datotekama.

Konfiguriranje aplikacije: aplikacija se konfigurira za rad na produkcijskom okruženju, a postavljaju se i svi potrebni parametri (npr. URL baze podataka, postavke sesije, sigurnosne postavke i slično).

Testiranje: nakon što je aplikacija postavljena na server, potrebno ju je temeljito testirati kako bi se osiguralo da sve funkcionalnosti rade ispravno i da nema grešaka.

Puštanje u rad: nakon uspješnog testiranja, aplikacija se pušta u rad i postaje dostupna korisnicima.

Ponovimo - pojmovi vezani za integraciju

- Što je to DNS?
- Podjela servera na shared, vps i dedicated.
- Prijenos projekta putem FTP-a.
- Prijenos projekta putem GIT-a.
- MySQL migracija.
- Kontinuirana integracija.
- Kontinuirani Razvoj.
- Neki od alata za kontinuiranu integraciju.

Što je to DNS?

- DNS (*Domain Name System*) je sustav koji omogućuje pretvaranje ljudski čitljivih imena domena u IP adrese koje koristi računalna mreža za komunikaciju.
- DNS funkcionira kao distribuirana baza podataka, gdje su podaci o nazivima domena raspoređeni u hijerarhijskoj strukturi DNS servera.
- DNS je ključan za rad interneta jer omogućuje da korisnici pristupe web stranicama koristeći njima razumljive nazive domena, umjesto da moraju zapamtiti IP adrese.

Podjela servera na shared, VPS i dedicated

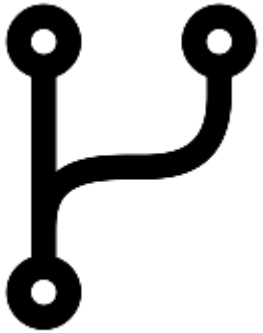
- **Shared hosting**
  - Shared hosting je vrsta hostinga u kojoj se više korisnika dijeli na istom fizičkom serveru, te svaki korisnik ima odvojeni dio resursa servera, ali koristi istu IP adresu i isti operativni sustav.
- **VPS hosting**
  - Virtual Private Server (VPS) je vrsta hostinga u kojoj se fizički server dijeli na virtualne servere, pri čemu svaki korisnik dobiva svoj odvojeni dio resursa.
- **Dedicated hosting**
  - Dedicated hosting je vrsta hostinga u kojoj se korisniku dodjeljuje cijeli fizički server, bez dijeljenja resursa s drugim korisnicima.

Prijenos projekta putem FTP-a

- FTP (*File Transfer Protocol*) je protokol za prijenos datoteka preko interneta.
- Prednosti FTP-a uključuju jednostavnost upotrebe i mogućnost prenosa velikih količina datoteka, kao i mogućnost pregleda i upravljanja datotekama na udaljenom serveru. FTP je također popularan i zbog svoje kompatibilnosti s većinom operativnih sustava.
- Međutim, FTP ima i neke nedostatke. Najveći problem s FTP-om je sigurnost, jer se korisničko ime i lozinka prenose preko mreže bez enkripcije, što znači da su osjetljive informacije poput lozinki vidljive drugim korisnicima na mreži. Alternativa FTP-u su sigurniji protokoli prijenosa datoteka kao što su SFTP (Secure File Transfer Protocol) i FTPS (FTP Secure), koji koriste enkripciju za sigurniji prijenos datoteka.

Izvor: Google Images

Prijenos projekta putem GIT-a



GIT je popularni sustav za upravljanje verzijama koda koji se koristi za suradnički razvoj softvera. Kada želite prenijeti projekt putem GIT-a, morate slijediti sljedeće korake:

Stvorite račun na GIT hosting platformi: Postoje različite platforme za GIT hosting, poput GitHuba, Bitbucket i GitLaba. Stvorite račun na odabranoj platformi i kreirajte novi repozitorij.

Inicijalizirajte GIT repozitorij na vašem lokalnom računalu: Pokrenite naredbu "git init" u direktoriju projekta na vašem lokalnom računalu. Ovo će stvoriti novi GIT repozitorij na vašem računalu.

Dodajte datoteke u repozitorij: Dodajte datoteke u novi GIT repozitorij pomoću naredbe "git add".

Napravite commit promjena: Koristite naredbu "git commit" za spremanje promjena u vašem lokalnom repozitoriju.

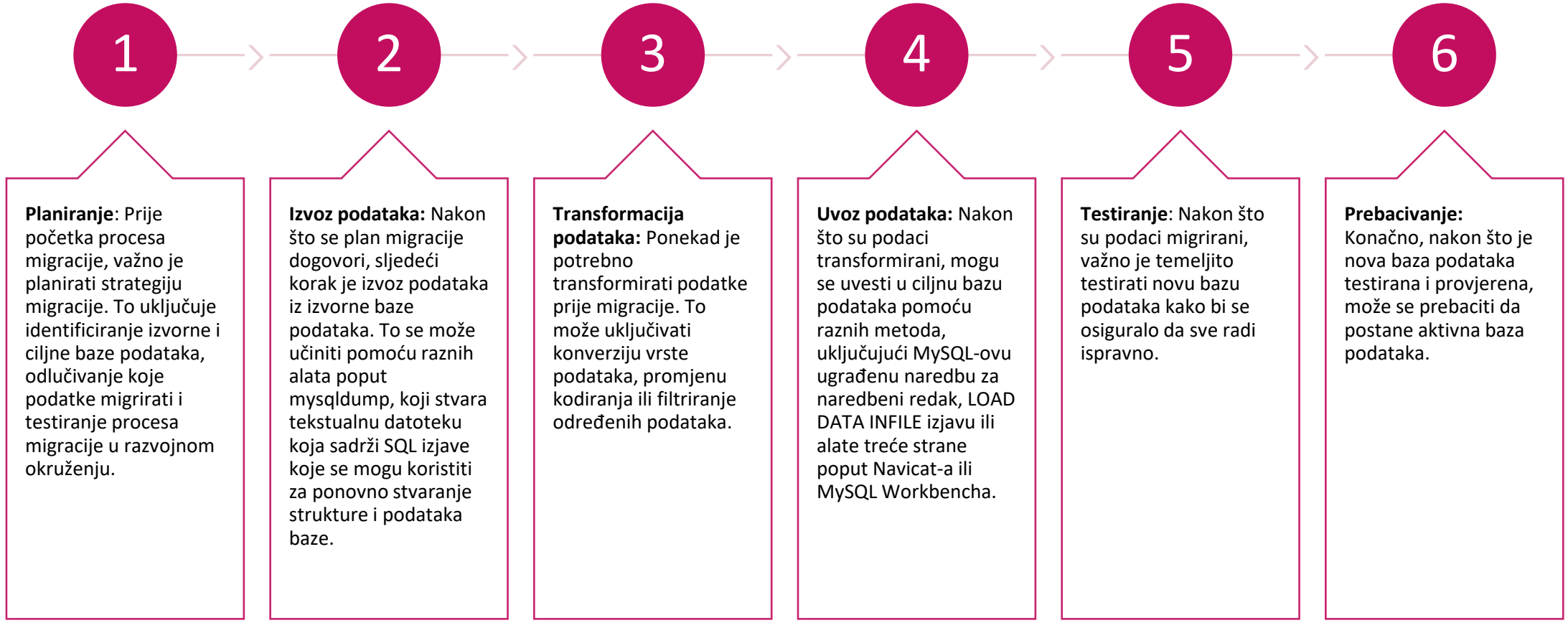
Dodajte udaljeni repozitorij: Dodajte udaljeni repozitorij pomoću naredbe "git remote add <naziv-repozitorija> <URL-repozitorija>". Ovo će omogućiti vašem lokalnom repozitoriju da se sinkronizira s udaljenim repozitorijem.

Pushajte promjene: Koristite naredbu "git push" za prenošenje vašeg lokalnog repozitorija na udaljeni repozitorij.

Nakon što ste uspješno prenijeli vaš projekt na GIT repozitorij, ostali članovi tima mogu povući najnoviju verziju projekta i nastaviti s radom na projektu.

MySQL migracija

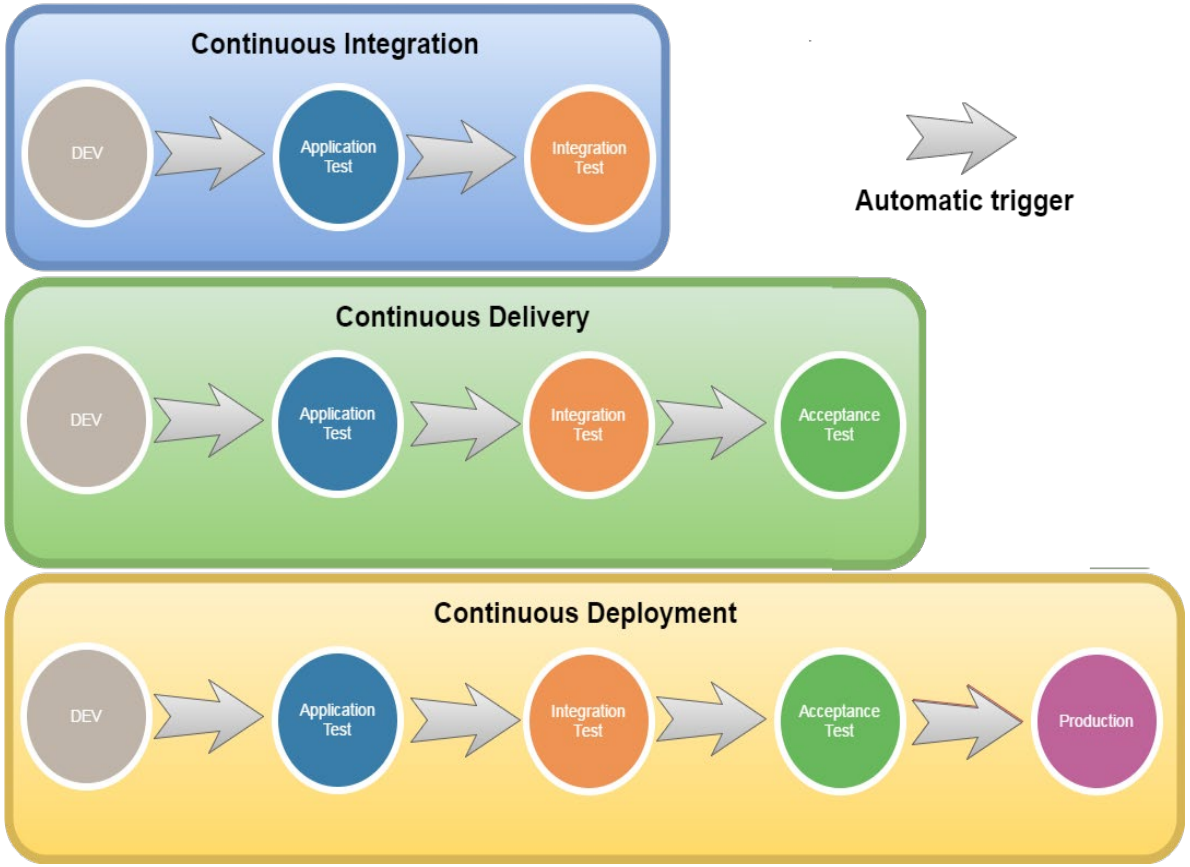
MySQL migracija je proces prebacivanja podataka s jedne MySQL baze podataka na drugu.



Kontinuirana integracija

- Kontinuirana integracija (eng. *Continuous Integration*, skraćeno CI) je praksa u softverskom razvoju u kojoj se često integriraju promjene u kodu u zajednički repozitorij, a nakon toga se automatski provode testovi kako bi se osiguralo da se novi kod integrira bez poteškoća i ne ometa postojeći kod.
- Proces kontinuirane integracije uključuje sljedeće korake:
  1. **Redovito ažuriranje zajedničkog repozitorija**: Razvojni timovi redovito ažuriraju zajednički repozitorij kako bi uključili nove promjene u kodu.
  2. **Automatsko generiranje izvršivih datoteka:** Nakon što su promjene u kodu uključene u zajednički repozitorij, sustav za kontinuiranu integraciju automatski generira izvršive datoteke, kao što su izvršne datoteke ili paketi za distribuciju.
  3. **Automatsko pokretanje testova:** Nakon generiranja izvršivih datoteka, sustav za kontinuiranu integraciju automatski pokreće testove kako bi se osiguralo da je novi kod integriran bez poteškoća i ne ometa postojeći kod.
  4. **Automatsko slanje obavijesti o rezultatima:** Nakon pokretanja testova, sustav za kontinuiranu integraciju šalje obavijesti razvojnom timu o rezultatima testova, uključujući i obavijesti o greškama koje su pronađene.

Kontinuirani razvoj



- Kontinuirani razvoj (eng. *Continuous Development*) je praksa u softverskom razvoju u kojoj se timovi fokusiraju na kontinuirano isporučivanje softverskih rješenja kroz automatizaciju procesa od planiranja, razvoja, testiranja do isporuke u produkciju.

Izvor: Google Images

Neki od alata za kontinuiranu integraciju

•**Jenkins:** Jenkins je open-source CI alat koji se može koristiti za automatiziranje procesa izgradnje, testiranja i isporuke softvera.
•**Travis CI:** Travis CI je popularni cloud-based CI alat za open-source projekte koji se izvode na GitHubu. Travis CI ima intuitivno sučelje i integrira se s velikim brojem različitih alata.
•**CircleCI:** CircleCI je cloud-based CI alat koji se može integrirati s različitim alatima za izgradnju, testiranje i isporuku softvera.
•**GitLab CI/CD:** GitLab CI/CD je integrirani CI/CD alat koji se koristi za automatiziranje procesa izgradnje, testiranja i isporuke softvera. GitLab ima podršku za kontejnere, a veliki broj funkcija su dostupni u besplatnoj verziji.
•**Bamboo:** Bamboo je CI alat koji se razvija od strane tvrtke Atlassian.

Izvor: Google Images

Ponovimo: Objasnite životni ciklus izdanja softvera.

Hvala na pažnji i sretno!