|  |
| --- |
| Avtor: Vid Kališnik |
| Razred: 4.B |
| Mentor: Vinko Kušar |

|  |
| --- |
| **MATURITETNA NALOGA** |
| IZDELAVA DINAMIČNE SPLETNE STRANI |



# Povzetek

V nalogi sem predstavil pridobivanje podatkov iz API-ja s pomočjo PHP-ja. Razložen je cel sistem skin-ov v igri Counter-Strike: Global Offensive, kako se cene razlikujejo in zakaj. Malo se tudi dotaknem podatkovnih baz (MySQL) in njuno uporabo v spletni strani. Spletna stran pa brez HTML, CSS in JavaScript-a nemore biti, zato je malo razlage še o tem.

I presented retrieving data from API using PHP. The whole system of skins in the game Counter-Strike: Global Offensive is explained, how prices differ and reasons why they differ. I briefly touch on databases (MySQL) and their use in my website. Main parts of the website are surely HTML, CSS and JavaScript, so I explain some uses of them.

## Ključne besede

Programiranje, algoritem, php, baze, sql, Counter-Strike: Global Offensive, API, JSON, cron-job, html, css, javascript, jquery, charts.js, cpanel

Kazalo

[Povzetek 1](#_Toc69224261)

[Ključne besede 1](#_Toc69224262)

[UVOD 4](#_Toc69224263)

[Opis informacijskega problema 4](#_Toc69224264)

[Namen spletne strani 4](#_Toc69224265)

[Razlika med dinamično in statično spletno stranjo 4](#_Toc69224266)

[Frontend in Backend 4](#_Toc69224267)

[BACKEND 5](#_Toc69224268)

[MySQL – Podatkovna baza 5](#_Toc69224269)

[E-R Model 5](#_Toc69224270)

[Razlaga E-R Modela 6](#_Toc69224271)

[PhpMyAdmin 6](#_Toc69224272)

[PHP – Koda na strežniku 7](#_Toc69224273)

[Povezava med PHP in MySQL 8](#_Toc69224274)

[Kaj je API 8](#_Toc69224275)

[Pridobivanje podatkov iz SteamAPI 9](#_Toc69224276)

[Izračun povprečij posameznih skin-ov 11](#_Toc69224277)

[Izračun povprečij zabjev (case-ov) 13](#_Toc69224278)

[Cron job 16](#_Toc69224279)

[FRONTEND 17](#_Toc69224280)

[HTML 17](#_Toc69224281)

[Glava 18](#_Toc69224282)

[Telo 18](#_Toc69224283)

[CSS 19](#_Toc69224284)

[Osnove 19](#_Toc69224285)

[Grid 20](#_Toc69224286)

[JavaScript 21](#_Toc69224287)

[jQuery 21](#_Toc69224288)

[Naslov do spletne strani 23](#_Toc69224289)

[Viri 23](#_Toc69224290)

Kazalo Slik

[Slika 1 E-R Model 5](#_Toc69315126)

[Slika 2 PhpMyAdmin 6](#_Toc69315127)

[Slika 3 Delovanje PHP 7](#_Toc69315128)

[Slika 4 Factory New Skin 11](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315129)

[Slika 5 Battle Scared Skin 11](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315130)

[Slika 6 Primer skin-ov v zaboju 13](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315131)

[Slika 7 Cron job 16](#_Toc69315132)

[Slika 8 HTML 17](#_Toc69315133)

[Slika 9 Osnove CSS 19](#_Toc69315134)

[Slika 10 Postavitev Grid (layout) 20](#_Toc69315135)

[Slika 11 jQuery - Galerija Slik 22](#_Toc69315136)

# Kazalo Kod

[Koda 1 Povezava med PHP in SQL 8](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315137)

[Koda 2 Include (php) 8](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315138)

[Koda 3 Pridobivanje podatkov iz podatkovne baze 9](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315139)

[Koda 4 SQL 9](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315140)

[Koda 5 Zamenjava znakov v string-u 10](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315141)

[Koda 6 Niz stanj skin-ov 10](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315142)

[Koda 7 Pridobivanje podatkov iz SteamAPI 10](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315143)

[Koda 8 Dva niza 11](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315144)

[Koda 9 For zanka 12](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315145)

[Koda 10 Zanka foreach in računanje povprečnih vrednosti 12](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315146)

[Koda 11 SQL - podatki iz dveh tabel 12](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315147)

[Koda 12 Izračun povprečja skin-a 12](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315148)

[Koda 13 Povprečna vrednost skin-a 13](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315149)

[Koda 14 Prirejanje nizov za kasnejšo uporabo 13](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315150)

[Koda 15 For zanka 14](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315151)

[Koda 16 For zanka in računanje povprečij za vsako barvo v zaboju 14](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315152)

[Koda 17 Zakluček računanja povprečja zaboja 15](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315153)

[Koda 18 Glava HTML 17](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315154)

[Koda 19 Odsek telesa HTML 18](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315155)

[Koda 20 Grid postavitev CSS 20](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315156)

[Koda 21 Določitev imena člena za grid 20](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315157)

[Koda 22 Prenos spremenljivke iz jQuery v PHP in povečanje zaslona o zaboju 21](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315158)

[Koda 23 Spremenljivka iz jQuery v PHP-ju 21](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315159)

[Koda 24 Gumba za naprej in nazaj 22](file:///C:\Users\mccha\Desktop\Maturitetna%20naloga\IZDELAVA%20DINAMICNE%20SPLETNE%20STRANI.docx#_Toc69315160)

# UVOD

## Opis informacijskega problema

Želimo narediti spletno stran, ki bo pridobivala podatke iz SteamAPI in z njimi računala. Ker bi podatke pridobivali avtomatsko bom uporabil cron-job. Koda pa bo napisana v PHP-ju, podatki bodo pa sharanjeni v MySQL bazi. Za oblikovanje spletne strani pa bomo uporabili tudi HTML, CSS in JavaScript (jQuery).

## Namen spletne strani

Pri računalniški igri Counter-Strike: Global Offensive lahko dobiš zaboje (case), ki pa jih lahko ali prodaš na tako imenovanem Community Market-u ali pa kupiš ključ za 2,5€ in zaboj odpreš. V njem lahko dobiš različno pobarvane puške (skin), nekatere so vredne 10 centov, nekatere so pa lahko več 100€. Vse skupaj je zelo podobno kockanju v casino-ju, a tukaj vse ceno “nastavi družba”. To pomeni, da cene niso že vnaprej določene in so odvisne od količine in povpraševanju po njih, kar pa pripeje do razlik v stopnji donosa za vsak zaboj.

Npr. Zaboj Prisma stane 4 cente, Chroma pa 70 centou torej, bi za prvega bi porabili 2,54€ za drugega pa 3,20€. Ker ima vsak zaboj različne skin-e in so ti razlčino vredni se stopnja donosa razlikuje. V času pisanja je stopnja donosa za zaboj Prisma 59,29%, Chroma pa 49,84%. Ker je zabojev 32 in se cene neprestano spreminjajo, je stopnjo donosa ze vsak zaboj nemogoče in nesmiselno “računati na roke” in zato rabimo napisati algoritem, ki bo vse to delal avtomatsko. S pomočjo algoritma pa bi radi ugotovili kateri zaboj se (po statistiki) najbolj splača odpreti.

## Razlika med dinamično in statično spletno stranjo

Glavna razlika je v tem, da dinamične spletne strani strežnik zgradi šele takrat, ko odjemalec zahteva spletno stran. Glede na zahtevo odjemalca ali glede na stanje podatkov na strežniku je lahko stran iz trenutka v trenutek drugačna. Zahtevo odjemalca strežnik posreduje programu, ki npr. nekaj izračuna, poišče podatke na strežniku, rezultate vgradi v spletni sestavek. Statične spletne strani so pa shranjene na spletnem strežniku. Ob vsaki zahtevi odjemalca, strežnik pošlje enako kopijo statične spletne strani. Statična spletna stran se spremeni le, ko upravitelj spremeni njene datoteke na stežniku.

## Frontend in Backend

Pri dinamični spletni strani delimo “delovno okolje” na dva dela: Frontend in Backend. Frontend je koda, ki se vidi na spletni strani torej HTML,CSS in Javascript. Backend pa je koda, ki je na spletnem brskalniku nemoremo videti – deluje v ozadju npr. PHP, NodeJS.

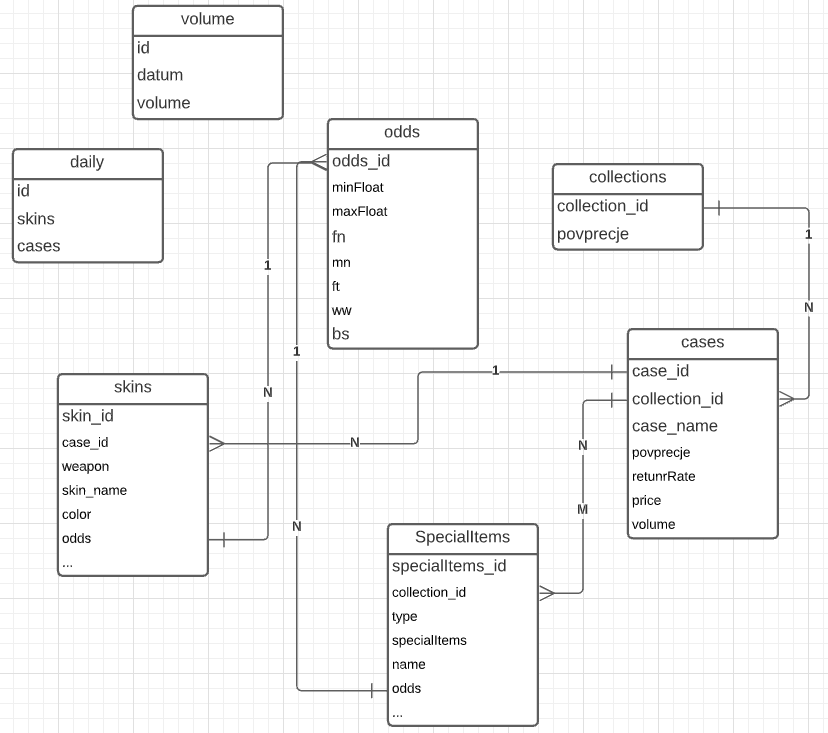
# [BACKEND](#_Frontend_in_Backend)

## MySQL – Podatkovna baza

MySQL je sistem za upravljanje s podatkovnimi bazami. MySQL je odprtokodna implementacija relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. MySQL deluje na principu odjemalec - strežnik, pri čemer lahko strežnik namestimo kot sistem, porazdeljen na več strežnikov.

### E-R Model

Pri spletni strani bom za boljši pregled naredil dve podatkovni bazi. V prvi bodo shranjeni podatki o skin-ih in razni izračuni. V drugi bo pa vsak case imel posebaj tabelo, zato da bodo podatki za nazaj tudi shranjeni, kar bo omogočalo risanje grafov. Torej so v drugi podatkovni bazi samo tabele brez relacij. Zato nas zanima E-R model prve.



Slika 1 E-R Model

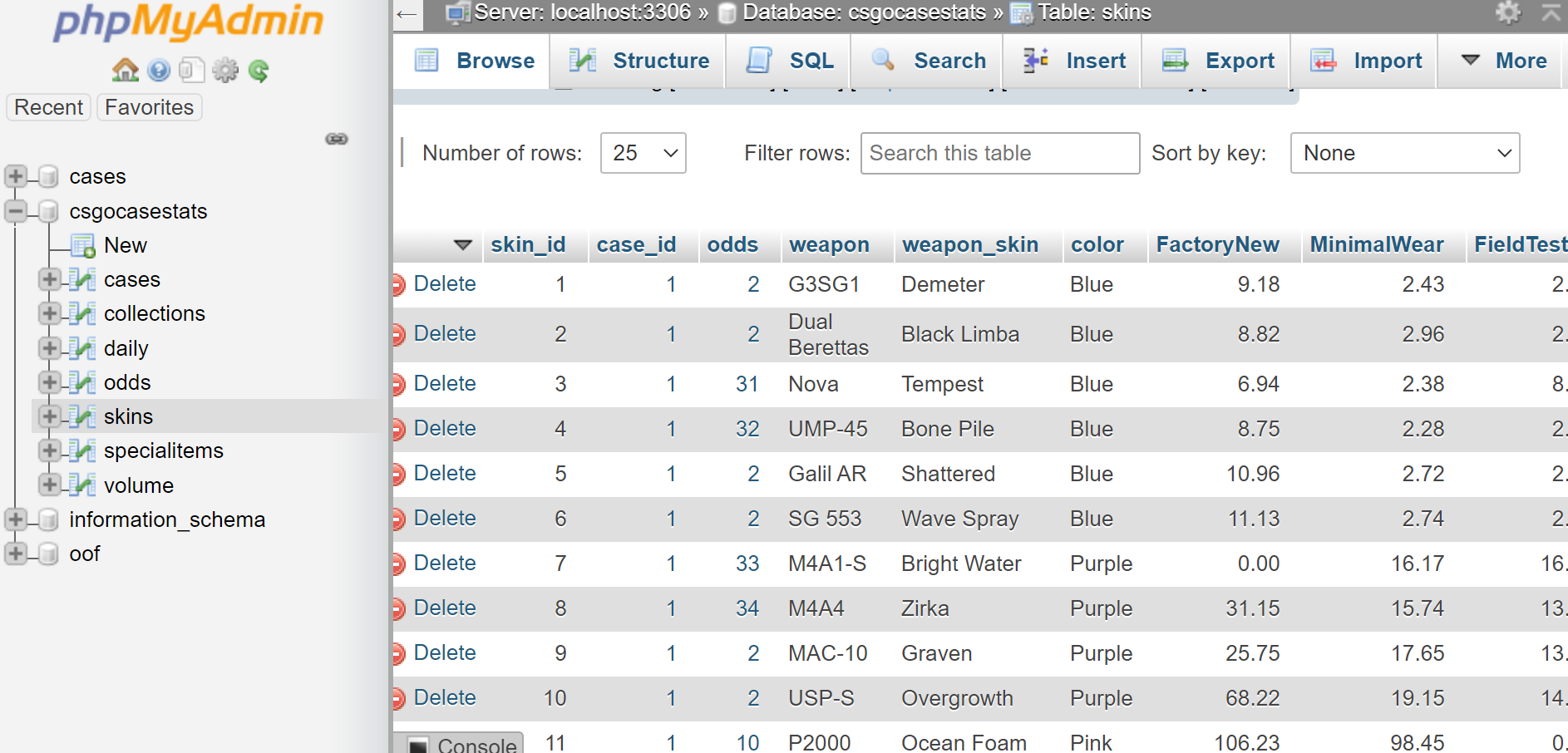
### Razlaga E-R Modela

Entiteta *cases* je v razmerju s entiteto *skins* z atributom case\_id, in sicer v razmerju 1:N, zato ker en zaboj (case) vsebuje več skin-ov, ti so pa lahko samo v enem zaboju. Podobno je razmerje z tabelo *SpecialItems* samo, da imajo lahko zaboji isto zbirko (collection) in je zato razmerje N:M.

Tabela *odds* pa je v istem razmerju 1:N s entiteto *skins* in *SpecialItems*. Razmerje je tako, ker ima vsak skin in specialItem samo ene možnosti koliko bo obrabrljen (wear), več skin-ov oz. specialItem-ov ima pa lahko iste možnosti.

### PhpMyAdmin

Podatkovno bazo in tabele v njej bom naredil v aplikaciji PhpMyAdmin, ki nam omogoča, da na preprost način urejamo podatkovno bazo.

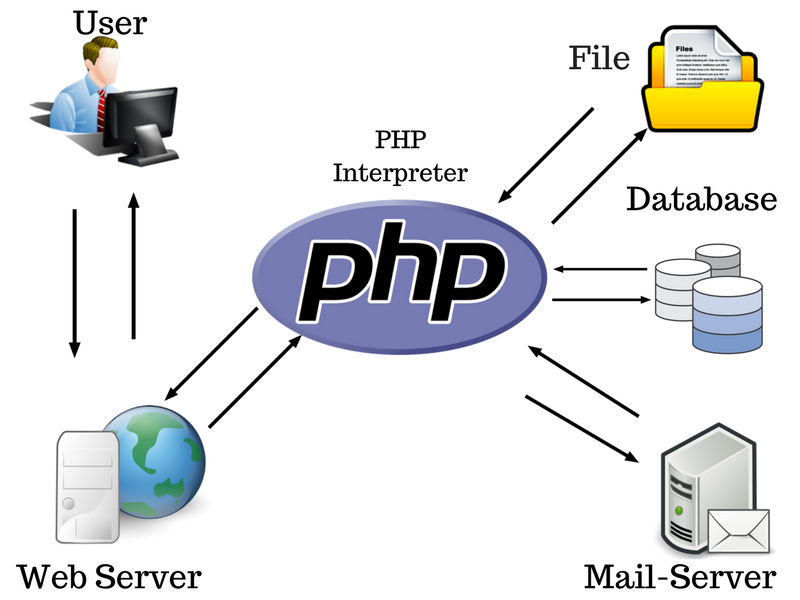


Slika 2 PhpMyAdmin

## PHP – Koda na strežniku

PHP (Hypertext Preprocessor) je razširjen odprtokodni programski jezik, ki se uporablja za razvoj dinamičnih spletnih vsebin. Je strežniški programski jezik, kar pomeni, da teče na strežniku, za njegovo delovanje pa potrebujemo spletni strežnik.

Spada v skupino interpreterskih programskih jezikov, kar pomeni, da se na začetku ne prevede celotna izvorna koda, ampak se sproti interpretira. Spletni strežnik ima to funkcijo, da interpretirano izvorno kodo pošlje brskalniku v obliki HTML kode. Tako uporabnik ne more videti izvorne kode.



Slika 3 Delovanje PHP

### Povezava med PHP in MySQL

Ker želimo s pomočjo PHP-ja pridobiti podatke in jih želimo shraniti v podatkovn bazo ju moramo povezati. Povežemo ju s pomočjo vgrajene funkcije mysqli (i je za improved).

Koda 1 Povezava med PHP in SQL

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "csgocasestats";

$dbname2 = "cases";

$conn = new mysqli($servername, $username, $password, $dbname);

$conn2 = new mysqli($servername, $username, $password, $dbname2);

if ($conn->connect\_error) {

  die("Connection failed: " . $conn->connect\_error);

};

?>

Spremenljivke se začnejo s znakom za dolar $, vrednost ji pa priredimo z enčajem (=). V funkcijo *new mysqli* mormo vnesti štiri parametre, ime strežnika, uporabniško ime, kodo in ime podatkovne baze iz katere želimo dobiti ali vnesti podatke. Funkcijo napišemo dvakrat, ker se bomo povezali s dvema podatkovnima bazama. S pogojnim stavkom *if* pa preverimo, če je povezava uspela. Če ni izpiše kaj je narobe.

Ker ne želimo povezave pisati pri vsaki datoteki, jo shranimo posebaj kot *conn.php* in pri drugih uporabimo funkcijo *include*.

Koda 2 Include (php)

include 'conn.php';

### Kaj je API

Vmesnik za namensko programiranje ali aplikacijski programski vmesnik (angleško application programming interface, API) je množica rutin, protokolov in orodij za izgradnjo programske opreme in aplikacij. API določa programsko komponento v smislu njenih operacij, izhodnih in vhodnih podatkov in z njimi povezanih tipov podatkov. API na ta način definira funkcionalnost programskih komponent na način, ki je neodvisen od dejanske implementacije. To omogoča različne implementacije komponente, ne da bi bil pri tem ogrožen predpisan vmesnik za uporabo komponente. API poenostavi razvoj programov s tem, da definira sestavne bloke, ki jih program sestavi pri izdelavi aplikacij. API lahko obstaja za mrežne sisteme, operacijske sisteme, podatkovne zbirke in programske knjižnice.

### Pridobivanje podatkov iz SteamAPI

SteamAPI nam omogoča pridobiti cene posameznih skin-ov. Podatke izpiše kot JSON.

JSON (JavaScript Object Notation) je preprost format za izmenjavo podatkov. Je enostaven za ljudi za branje in pisanje in tudi za računalnike za razbiranje in generiranje.

Primer:

<https://steamcommunity.com/market/priceoverview/?appid=730&currency=1&market_hash_name=AWP%20%7C%20Atheris%20%28Field-Tested%29>

(sivo pobarvan del je za vse enak)

Ob kliku na link se nam izpiše:

{"success":true,"lowest\_price":"$3.05","volume":"1,778","median\_price":"$2.91"}

Zaradi omejenega števila prošenj (request) mormo med vsako prošenjo počakati. Zato je v prvi podatkovni bazi *csgocasestats* entiteta *daily*, v kateri je atribut *skins*, ki nam pove ID skina, ki je na vrsti. Torej iz podatkovne baze prebremo ID skin-a.

V spremenljivko *$sql* vpišemo kodo SQL, ki v tabli *daily* poišče atribut *skins*. Taki kodi se reče query, po slovensko poizvedba. Nato v funkcijo *mysqli\_query­* vnesemo dva parametra, prvi je povezava do podatkovne baze, drugi pa je koda SQL. Ker pa je rezultat shranjen kot SQL poizvedba jo moramo še pretvoriti v besedilo (string). V ta namen uporabimo funkcijo mysqli\_fetch\_array(), ki za argument sprejme rezultat naše MySQL poizvedbe in vrne eno-dimenzionalno polje s trenutno vrstico rezultatov. PHP nima funkcije, ki bi poizvedbo takoj pretvoril v besedilo. Zato nakoncu spremenimo niz (array) v besedilo tako, da samemu sebi priredi prvo vrednost.

Koda 3 Pridobivanje podatkov iz podatkovne baze

$sql = "SELECT skins FROM daily";

$result = mysqli\_query($conn, $sql);

$skinId = mysqli\_fetch\_array($result);

$skinId = $skinId[0];

Za ime pištole in skin-a naredimo podobno samo zamenjamo iz katere tabele bomo vzeli podatke in v SQL kodo dodamo WHERE, ki vzame samo podatke kjer je določen *skin\_id* v našem primeru bomo vzeli spremenljivko, ki smo jo dobili prej.

Koda 4 SQL

$sql = "SELECT weapon FROM skins WHERE skin\_id=$skinId;";

Ostalo koda za pridobitev podatkov iz podatkovne baze je ista kot pri prejšnjem primeru. Sedaj imamo tip orožja in rabimo samo še ime skin-a, ki ga dobimo na isti način samo, da vzamemo iz tabele *skins*.

Ker v URL nemoremo pisati presledkov mormo vse podatke, ki smo jih vzeli iz podatkovne baze urediti tako, da znake, ki jih v URL nemoremo zapisati spremenimo.

Koda 5 Zamenjava znakov v string-u

$zamenjava = array(" " => "%20","'" => "%27", "(" => "%28", ")" => "%29" );

$skupi = strtr($weapon,$zamenjava)."%20%7C%20".strtr($weaponSkin,$zamenjava);

Zato naredimo associative array (niz v katerem so podatki poimenovani) in vanj vpišemo kaj želimo zamenjati s čim. Na primer ( bomo zamenjali z %28, vsak space z %20 in tako dalje. Spremanljivka *$skupi* združi ime pištole in skin pištole. V funkcijo *strtr* vpišemo dva parametra. Prvi je tisti pri katerem želimo nekaj zamenjati, drugi pa s čim bi zamenjali.

Če še enkrat pogledamo kakšen URL želimo dobiti, imamo že skoraj vse. Potrebujemo samo še zadnji del, pri katerem je pa za vsak skin pet enakih možnosti. Zato naredimo niz v katerega vse vpišemo.

<https://steamcommunity.com/market/priceoverview/?appid=730&currency=1&market_hash_name=AWP%20%7C%20Atheris%20%28Field-Tested%29>

Koda 6 Niz stanj skin-ov

$conditionsURL =["%20%28Factory%20New%29", "%20%28Minimal%20Wear%29","%20%28Field-Tested%29","%20%28Well-Worn%29","%20%28Battle-Scarred%29"];

Vsak skin ima 5 vrst koliko je obrabljen (condition). Zato rabimo dobiti ceno vsakega in naredimo zanko *for,* ki se ponovi petkrat. Pri spremenjlivki *$filename* povežemo vse, kar rabimo, da nam URL deluje. S funkcijo ***file\_get\_contents*** pridobimo podatke iz strani. Ker pa so podatki v JSON formatu jih moramo pretvoriti v PHP array to pa nam omogoča funkcija ***json\_decode***. Rabimo samo podatke *lowest\_price* zato spremenljivki *$lowest\_price*  priredimo to vrednost. Ker želimo s to ceno še računati jo moramo v podatkovno bazo shraniti kot *float*, zato s funkcijo ***strtr*** odstrsanimo znak dolar.

Koda 7 Pridobivanje podatkov iz SteamAPI

for($i=0;$i<5;$i++){

        $filename     =       "https://steamcommunity.com/market/priceoverview/?appid=730&currency=1&market\_hash\_name=".$skupi.$conditionsURL[$i];

        $data         =       file\_get\_contents($filename);

        $array        =       json\_decode($data, true);

        $lowest\_price =       $array["lowest\_price"];

        $d1        =       strtr("$lowest\_price","$"," ");

//posodobi  ceno

        $sql = "UPDATE skins SET $conditions[$i]='$d1' WHERE skin\_id=$skinId";

        if ($conn->query($sql) === TRUE) {

          echo "Record updated successfully";

        } else {

          echo "Error updating record: " . $conn->error;

        };

};

Na koncu moramo samo še podatke vnesti v podatkovno bazo. Ker podatke želimo samo posodobiti uporabimo UPDATE. Nato napišemo v katero tablo, kateri stolpec in kateri podatek in še nazadnje še v kateri vrsti želimo podatke posodobiti. Nato s *if* preverimo, če je uspešno posodobilo podatke.

Podobno kodo uporabimo tudi za pridobitev cene in števila prodanih v zadnjih 24 urah zabojev (case-ov).

### Izračun povprečij posameznih skin-ov

Kateri koli skin ima neko vrednost med 0 in 1 (rečemo float), ki določa koliko je skin obrabljen. Vsi skini pa imajo pet intervalov katere za lažjo predstavo poimenujemo :

* 0 – 0,07 (3%)-> Factory New ( najmanj obrabljen, te so večinoma najdražji)
* 0,07 – 0,15 (24%)-> Minimal Wear
* 0,15 – 0,38 (33%)-> Field-Tested
* 0,38 – 0,45 (24%)-> Well-Worm
* 0,45 – 1 (16%)-> Battle Scared (najbolj obrabljeni)



Slika 4 Factory New Skin

*Zanimivost: Cena zgornjega skin-a je nekje med 2000 in 5000€*

Slika 5 Battle Scared Skin

Ker pa ima vsak skin posebej določen interval (nekje od 0-1) v katerem lahko ta skin dobimo so verjetnosti za vse različne. Na primer pri skinu, ki ga lahko dobimo samo od 0,06 do 0,8 je verjetnost, da skin dobimo v Factory New 0,58%. Zato moramo iz podatkovne baze vzeti ceno vsakega stanja skin-a in kolikšna je verjetnost zanj.

Za začetek naredimo dva niza (array-a), da ju bomo lahko uporabili v zanki *foreach*. V nizih so imena stolpcev, iz katerih bomo vzeli podatke.

Koda 8 Dva niza

$conditions = ["FactoryNew","MinimalWear","FieldTested","WellWorm","Battled"];

$conditionsSt = ["FactoryNewSt","MinimalWearSt","FieldTestedSt","WellWormSt","BattleScaredSt"];

Vseh skin-ov v vseh zabojih je 521, zato naredimo zanko *for*, ki bo kodo ponovila 521-krat.

for($i=1;$i<522;$i++){

Koda 9 For zanka

Nato znotraj zanke naredimo še dve *foreach* zanki, eno za navadne cene skin-ov drugo pa za StatTrack skine. Ker sta obe isti, razlika je samo v tem da pri prvi uporabimo zgoraj navedeni prvi niz v drugi pa drugega.

Koda 10 Zanka foreach in računanje povprečnih vrednosti

foreach($conditions as $condition){

$sql = "SELECT $condition FROM skins WHERE skin\_id='$i';";

$result = mysqli\_query($conn, $sql);

$cena = mysqli\_fetch\_array($result);

$sql = "SELECT  o.$condition FROM skins s JOIN odds o ON s.odds = o.odds\_id  WHERE skin\_id='$i'";

$result = mysqli\_query($conn, $sql);

$verjetnost = mysqli\_fetch\_array($result);

$navadni = $cena[0]\*$verjetnost[0];

$navadniAvg += $navadni;

};

V zanko *foreach* vpišemo kateri niz želimo uporabiti, nato sledi *as* (kot) in napišemo kako bo ime posameznih podatkov v nizu. Sledi ista koda, ki smo jo že uporabili za pridobitev podatkov iz podatkovne baze. Sedaj imamo ceno in rabimo samo še podatek kolikšna je verjetnost za to stanje skin-a. Ker so ti podatki shranjeni v drugi tabeli moramo SQL kodo nadgraditi:

Koda 11 SQL - podatki iz dveh tabel

SELECT  o.$condition

FROM skins s // tabela skins – v njej so cene skin-ov

JOIN odds o // tabela odds – v njej so kvote za vsako stanje

ON s.odds = o.odds\_id // ta ukaz poišče kakšen odds.id ima skin, zato da lahko iz tabele odds vzame prave kvote

WHERE skin\_id='$i'

Z zmožkom spremenljivke *$cena* in *$verjetnost* dobimo povprečje enega od stanj skina, zato jo prištejemo h spremenljivki *$navadniAvg*. Ko gre zanka čez petkrat dobimo povprečno vrednost skin-a.

Koda 12 Izračun povprečja skin-a

$navadni = $cena[0]\*$verjetnost[0];

$navadniAvg += $navadni;

Po dveh zankah *foreach* dobimo vrednost *$navadniAvg* (povprečje cen navadnega skin-a) in *$stAvg* (povprečje cen StatTrack). Ker pa je možnost za StatTrack samo 10% jo pomnožimo z 0,1 , navadne pa z 0,9.

Koda 13 Povprečna vrednost skin-a

 $skinAvg = $navadniAvg\*0.9 + $stAvg\*0.1;

    $sql = "UPDATE skins SET povprecje='$skinAvg' WHERE skin\_id='$i'";

    if ($conn->query($sql) === TRUE) {

      echo "Record updated successfully". "<br>";

    } else {

      echo "Error updating record: " . $conn->error;

    };

    $navadniAvg = 0;

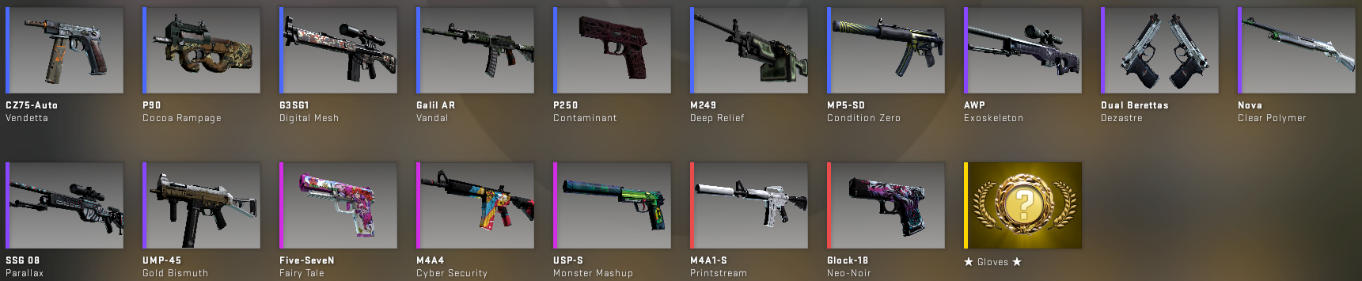
    $stAvg = 0;

}; // tukaj se šele zapre zanka for

Na koncu posodobimo povprecje v tabeli skins in vrednosti *$navadniAvg* in *$stAvg* nastavimo na 0 za naslednje skine, ker če tega nebi storili bi se povprečje prejšnjega seštelo z novim itd.

### Izračun povprečij zabjev (case-ov)

V vsakem zaboju so skini razdeljeni v pet skupin:

* Modre (79.92%) – so najbolj pogosti
* Vijolične (15.98%)
* Rozne (3.19%)
* Rdeče (0.64%)
* Rumene (0.26%) – nož ali rokavice (nekateri zaboji imajo iste)

Slika 6 Primer skin-ov v zaboju

Zato je v tabeli *skins* stolpec color, kjer je za vsak skin določena barva. V kodi nato naredimo dva niza, ki ju bomo uporabili kasneje. Spremenljivka *$len* pa bo štela koliko skin-ov iste barve je v zabojih (ker je v vsaki različno število).

Koda 14 Prirejanje nizov za kasnejšo uporabo

$colors = ['Blue','Purple','Pink','Red'];

$colorsProcent = [0.7992,0.1598,0.0319,0.0064,0.0026];

$len = 0;

Enako kot prejšnič naredimo ***for***zanko, ki bo šla skozi vse zaboje.

Koda 15 For zanka

for($i=1;$i<33;$i++){ //za case

Za njo pa napišemo še eno ***for***zanko, ki bo izračunala povprečja za vsako barvo posebaj.

Koda 16 For zanka in računanje povprečij za vsako barvo v zaboju

for($j=0;$j<4;$j++){    // za vse barve

   $result = mysqli\_query($conn,

   "SELECT povprecje FROM skins  WHERE case\_id = $i AND color = '$colors[$j]'"

   );

   while($row = $result->fetch\_assoc()) {

       ++$len;

       $povprecje += $row["povprecje"];

   };

   $povprecje = $povprecje / $len;

   if($j == 0){

            $blueAvg = $povprecje\*$colorsProcent[0];

   }elseif($j == 1){

            $purpleAvg = $povprecje\*$colorsProcent[1];

   }elseif($j == 2){

            $pinkAvg = $povprecje\*$colorsProcent[2];

   }elseif($j == 3){

            $redAvg = $povprecje\*$colorsProcent[3];

   };

   $len = 0;

   $povprecje = 0;

};

Najprej iz podatkovne baze vzamemo vsa povprečja skin-ov (te smo izračunali v prejšnjem sklopu), ki imajo isti *case\_id* (torej so iz istega zaboja) in isto barvo. Ker pa je takih podatkov več kot en moramo napisati še ***while***zanko, katera gre skozi vse rezultate. Znotraj zanke povečujemo spremenljivko *$len,* da izvemo koliko takih podatkov smo dobili, in nato še spremenljivki *$povprecje* prištejemo vsa povprečja skin-ov. Nato izven zanke *while* spremeljivko *$povprecje* delimo s spremenljivko *$len*, da dobimo povprečje vseh skin-ov te barve. Temu sledi funkcija *if*, ki preveri povprečje katerih barv smo izračunali in ga množi s njegovo verjetnostjo. Na koncu še za naslednje izračune nastavimo spremeljivki *$len* in *$povprecje* na 0 in zapremo zanko *for.*

Po zanki *for,* ki nam izračuna povprečja za vsako barvo, razen rumene (izračunamo jo podobno kot skin-e), ki pa ga dobimo tako, da najprej vzamemo podatke kateri *collection­\_id* ima case in nato iz tabele kjer so izračunana povprečja za vsak *collection* vzamemo tistega, ki ima isti *collection­\_id*  kot zaboj. Potem to povprečje pomnožimo s spremenljivko *$colorsProcent[4].* Sedaj samo še vsa povprečja seštejemo in dobimo koliko denarja bi povprečno dobili, če bi odpreli zaboj.

Koda 17 Zakluček računanja povprečja zaboja

    $collection = mysqli\_fetch\_array(mysqli\_query($conn,

        "SELECT collection\_id FROM cases WHERE case\_id = $i"

    ));

    $collectionAvg = mysqli\_fetch\_array(mysqli\_query($conn,

      "SELECT povprecje FROM collections WHERE collection\_id = $collection[0]"

    ));

    $collectionAvg = $collectionAvg[0]\*$colorsProcent[4];

    $caseAvg = $blueAvg + $purpleAvg + $pinkAvg + $redAvg + $collectionAvg;

    echo $caseAvg."<br>";

    $sql = "UPDATE cases SET povprecje='$caseAvg' WHERE case\_id='$i'";

    if ($conn->query($sql) === TRUE) {

      echo "Record updated successfully". "<br>";

    } else {

      echo "Error updating record: " . $conn->error;

    };

};

### 

### Cron job

Cron job nam omogoča samodejno izpeljavo programov na strežniku. Torej se lahko zgoraj opisani programi za pridobivanje podatkov izpeljejo sami in tako podatke osvežujejo. Dobra stran cpanel-a je da je cron job vanj že vgrajen. Tako moramo vpisati samo kdaj, kolikokrat in kateri program naj se izvede.



Slika 7 Cron job

Prve dva ukaza delujeta vsako uro (razen eno) in na 20 minut. Tak razpon med vsako ponovitvijo je pa zato, ker če prevečkrat uporabimo SteamAPI neha delovati. Vmes pa eno uro nedelata, ker takrat deluje zadnji ukaz. Tretji (izračuna povprečje skin-ov) in četrti (izračuna povprečje case-ov) ukaz pa delujeta vsako minuto in vsako uro, ker samo računata. Na koncu vsakega ukaza pa napišemo >/dev/null 2>&1 , da ne dobivamo ob koncu izpis vsakega programa na e-mail.

# FRONTEND

## HTML

HTML (HyperText Markup Language) je jezik, s katerim opišemo gradiva, ki jih želimo objaviti na spletu. Popularen je postal kmalu po svojem nastanku leta 1990, od takrat pa je doživel že precej sprememb. Njegov razvoj so spremljala številna nesoglasja med večjimi programerskimi hišami, ki so kar tekmovale, kdo bo ponudil boljši program (brskalnik) za pregledovanje tako opisanih gradiv. K sreči je HTML zdaj standardiziran, še vedno pa si brskalniki kakšno stvar iz standarda razlagajo po svoje, tako da imamo včasih kar nekaj težav, ko želimo spletno stran napisati tako, da bo v vseh brskalnikih izgledala enako.



Slika 8 HTML

### Glava

Glava HTML dokumenta lahko vsebuje določene oznake, skripte, meta podatke... V glavi med oznakama <head> in </head> opišemo dokument, pri tem lahko dodamo naslednje oznake: title, base, link, meta, script in style. Kar napišemo v glavi, v oknu brskalnika ni vidno.

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Csgo Case Stats</title> <!-- Naslov spletne strani -->

    <link rel="icon" href="images/logo.png"> <!-- Logotip spletne strani -->

    <link rel="stylesheet" href="./css/style.css"><!—Povezava s css datoteko->

    <!— Povezava s jQuery in Charts.js  -->

<script src="https://code.jquery.com/jquery-3.5.1.js" integrity="sha256-QWo7LDvxbWT2tB53yJnYU3WhH/C8ycbRAkjPDc=" crossorigin="anonymous"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js" integrity="sha512-d9xgZrVZpmm5qylQ/yWlFAyY6hYgdF1Qh6nYiuADWwKB4C2WSw==" crossorigin="anonymous"> </script>

</head>

Koda 18 Glava HTML

### Telo

Drugi del HTML dokumenta se imenuje BODY. V tem delu je oblikovan dokument, ki ga pregledujemo s pregledovalnikom spletnih strani. Sama vsebina dokumenta je zaprta med oznaki <body> in </body>.

Koda 19 Odsek telesa HTML

<!-- NAJBOLJŠI RETURN RATE BOX -->

<div class="case1 flex-column expand zgornji">

    <a href="caseInfo.php" class="together">

         <div class="black-box bb-small shadow center">

             <h3 class="black-box-text">Best Return Rate</h3>

         </div>

     <div class="white-box wb-small shadow">

         <div class="flex-center">

          <?php include 'phps/sort.php'; echo   '<img src="images/'.$caseFolder\_rR.'/256fx256f.png" alt="" class="img-center"><h2 class="case-name">'.$caseName\_rR.'</h2>';?>

          </div>

     </div>

    </a>

</div>

Značka **div** nam omogoča, da kodo razdelimo na posamezne oddelke, kar je še posebaj koristno pri dodajanju CSS-a. V znački div dodamo še **class,** s katerim lahko nato izberemo prav ta oddelek. Značka **a** deluje kot povezava med drugimi datotekami. V zgornji kodi je napisana tudi koda v PHP-ju, ker se prikazani zaboji spreminjajo glede, na razne kriterije. Torej vse kar naredimo je, da izpišemo HTML kodo v kateri so spremenljivke, ki pa se v končni kodi na spletu vidijo samo kot besedilo.

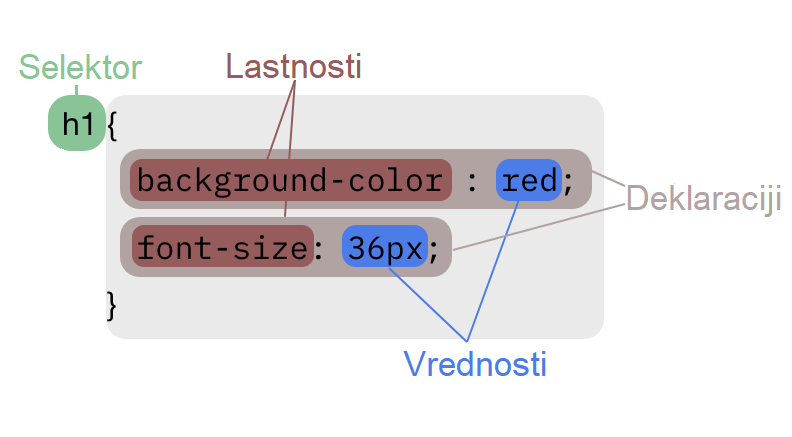
## CSS

Cascading Style Sheets (sl. kaskadne stilske podloge) poznane pod kratico CSS so podloge, predstavljene v obliki preprostega slogovnega jezika, ki skrbi za prezentacijo spletnih strani. Z njimi definiramo stil HTML oz. XHTML elementov v smislu pravil, kako se naj ti prikažejo na strani. Določamo lahko barve, velikosti, odmike, poravnave, obrobe, pozicije in vrsto drugih atributov, prav tako pa lahko nadziramo aktivnosti, ki jih uporabnik nad elementi strani izvaja (npr. prekritje povezave z miško). Podloge so bile razvite z namenom konsistentnega načina podajanja informacij o stilu spletnim dokumentom.

Bistvo uporabe CSS je poleg defirniranja pravil, pred vsem ločitev strukture strani - ki jo podaja označevalni jezik skupaj z vsebino - od njene predstavitve. S tem omogočimo lažje urejanje in dodajanje stilov ter poskrbimo za večjo preglednost dokumentov temelječih na HTML sintaksi. Prav tako zmanjšamo ponavljanja kode, saj omogočimo množici strani uporabo istih podlog, kar lahko bistveno zmanjša njihovo velikost.

### Osnove

Slogovna predloga je zgrajena iz niza pravil. Pravilo je sestavljeno iz dveh delov: iz *selektorja* in iz *deklaracije*. Deklaracija je sestavljena iz lastnosti in vrednosti.



Slika 9 Osnove CSS

### Grid

S pomočjo ***grid*** postavitve lahko vse html elemente uredimo tako kot si želimo, brez velikih naporov.

V HTML dokumentu je telo v oddelku (div), ki ima razred *container*. Določimo mu višino (height) in sicer vh pomeni *viewheight* (vidna višina), torej če damo 100 bo zavzel celo vidno območje.

Koda 20 Grid postavitev CSS

.container{

    display: grid; /\* določimo postavitev grid \*/

    height: 100vh;

    grid-template-columns: 1.93fr 2.63fr 2.63fr 2.81fr; /\* določimo koliko in velikost stolpcev\*/

    grid-template-rows: 2fr 2fr; /\* določimo koliko in velikost vrstic \*/

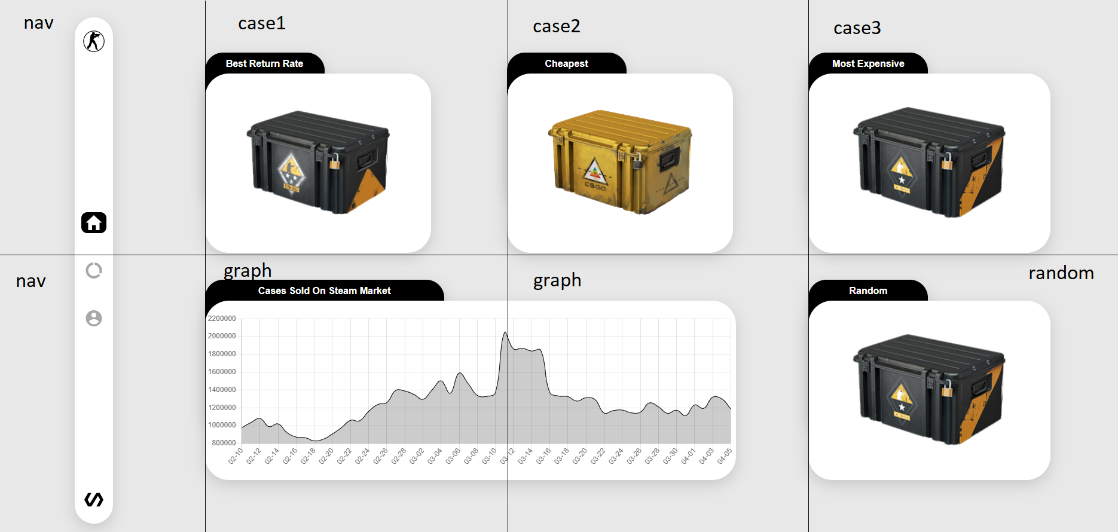
    grid-template-areas:

        "nav case1 case2 case3"

        "nav graph graph random";

}

V polje *grid-template-areas* moramo vpisati kako bodo elementi razporejeni. Ker imamo štiri stolpce, moramo v prvi vrstici napisati štiri člene, in enako za drugo vrstico. Če člen napišemo v dveh delih grid-a bo zavzemal prostor obeh.



Slika 10 Postavitev Grid (layout)

Ime člena pa določimo tako, da izberemo razred elementa in mu v CSS kodi napišemo:

Koda 21 Določitev imena člena za grid

.case1{

    grid-area: case1;

}

## JavaScript

JavaScript je objektni skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani. Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko sodeluje s HTML-kodo in s tem poživi stran z dinamičnim izvajanjem. JavaScript podpirajo velika programska podjetja in kot odprt jezik ga lahko uporablja vsakdo, ne da bi pri tem potreboval licenco. Podpirajo ga vsi novejši spletni brskalniki.

### jQuery

jQuery je knjižnica z JavaScript funkcijami, ki omogočajo: izbiro in manipulacijo s HTML elementi, manipulacijo z dogodki na spletni strani, premikanje po HTML DOM modelu, manipulacijo z njegovimi objekti in z elementi jezika CSS, uporabo že pripravljenih efektov in animacij,asinhrono komunikacijo med odjemalcem in strežnikom (AJAX) in različna druga podporna orodja.

#### AJAX

Ajax (asinhroni JavaScript in XML) je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij. Z Ajaxom si lahko spletne aplikacije izmenjujejo podatke s strežnikom asinhrono v ozadju, brez potrebe po ponovnem nalaganju strani.

Z uporabo AJAX-a se bo ob kliku na posamezen zaboj odprelo novo okno z večjimi podatki o zaboju. Vsi ukazi za delo s knjižnico jQuery so navadno vključeni v ukaz, ki preprečuje izvajanje jQuery funkcij, preden se spletna stran ne naloži v celoti (prva vrstica). V drugi vrstici določimo, da ko kliknemo na element s razredom *case1* se izvede funkcija. V kateri določimo spremenljivko *case1* in sicer ji določimo enako vrednost, ki je znotraj HTML elemnenta s razredom *case1* in *case-name*. Nato pa celo polje s razredom *replace* zamenjamo s kodo napisano v datoteki *caseInfo*.*php*, poleg tega pa določimo ime spremenljivke *caseName* in njeno vrednost *case1*, ki smo jo določi v tretji vrstici.

Koda 22 Prenos spremenljivke iz jQuery v PHP in povečanje zaslona o zaboju

1 $(document).ready(function(){

2        $('.case1').click(function(){

3            var case1 = $('.case1 .case-name').text();

4            $(".replace").load("./phps/caseInfo.php", {

5                caseName: case1,

6            });

7        });

8 });

Spremenljivko *caseName* pa nato uporabimo v PHP-ju s pomočjo funkcije \_POST.

Koda 23 Spremenljivka iz jQuery v PHP-ju

$\_POST['caseName'];

#### Galerija slik

Naš clij je, da ob kliku na eno od puščic (levo, desno), se nam pokaže naslednja slika in tabela, v kateri so podatki o skin-u na sliki.



Slika 11 jQuery - Galerija Slik

Najprej določimo, da se ob kliku na vsak element v HTML kodi, ki ima razred *next* izvede funkcija (vrstica 2). Nato v funkciji določimo dve spremenljivki, prva *current* izbere HTML element, s razredom *active*, pri drugi *next* pa uporabimo funkcijo *next*, ki izbere nasledni element od elementa, ki smo ga podali. Za tem napišem pogojni stavek *if,* ki se izvede, če obstaja nasledni element. Torej, če obstaja damo naslednjem elementu razred *active* prejšnjemu pa razred odstranimo. Enako je za gumb za nazaj, samo da namesto *next* vpišemo *prev.*

Koda 24 Gumba za naprej in nazaj

1 $(document).ready(function() {

// Gumb za naprej

2    $(".next").on('click', function(){

3        var current = $('.active');

4        var next = current.next();

5

6        if(next.length){

7           current.removeClass('active').css('z-index','-10')

8           next.addClass('active').css('z-index','10')

9        }

10    });

11 // Gumb za nazaj

12    $(".prev").on('click', function(){

13        var current = $('.active');

14        var prev = current.prev();

15

16        if(prev.length){

17            current.removeClass('active').css('z-index','-10')

18            prev.addClass('active').css('z-index','10')

19        }

20     });

21 });

# Naslov do spletne strani

<http://csgocasestats.com/>

# Viri

**2020.** A Technical Look at How CS:GO Float Values Work. *csgofloat.* [Elektronski] 2020. https://blog.csgofloat.com/how-csgo-float-values-work/.

**2021.** AJAX. *jQuery.* [Elektronski] 2021. https://api.jquery.com/jquery.ajax/.

**2021.** Charts.js. [Elektronski] 2021. https://www.chartjs.org/docs/latest/.

**Jennifer Kyrnin, Julie C. Meloni. 2019.** *Sams teach yourself HTML, CSS, and JavaScript all in one.* New York : Pearson Education, 2019. 978-0-672-33808-3.

**2019.** jQuery. *YouTube.* [Elektronski] 2019. https://www.youtube.com/watch?v=ScoURsEM\_yU.

**Krebelj, Peter. 2015.** *HTML in CSS za začetnike.* Ljubljana : Atelje Doria, 2015. 978-961-6881-35-7.

**2021.** PHP. *Wikipedia.* [Elektronski] 2021. https://en.wikipedia.org/wiki/PHP.

**2020.** PhpMyAdmin. *YouTube.* [Elektronski] 2020. https://www.youtube.com/watch?v=tz5iWA9tbpE.

**Štrancar, Matjaž in Klemen, Simon. 2005.** *PHP in MySQL na spletnem strežniku Apache / Matjaž Štrancar in Simon Klemen.* Ljubljana : Pasadena, 2005. 961-6361-53-8.

**2021.** W3Schools. [Elektronski] 2021. https://www.w3schools.com/.