# Web aplikacija za prikupljanje i prikaz podataka s IoT stanice

**Kopjar, Bernardo**

**Undergraduate thesis / Završni rad**

**2021**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:202789

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-18**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Stručni studij informatike**

# WEB APLIKACIJA ZA PRIKUPLJANJE I PRIKAZ PODATAKA S IOT STANICE

**Završni rad**

**Bernardo Kopjar**

**Osijek, 2021.**

**SADRŽAJ**


1. UVOD ..... 1
   1.1. Zadatak završnog rada ..... 1
2. POSTOJEĆE APLIKACIJE ..... 2
   2.1. Cras Telecontrol Service ..... 2
   2.2. Humidex myHome aplikacija ..... 2
   2.3. EDGEVue web aplikacija ..... 3
3. WEB APLIKACIJA ..... 5
   3.1. Kako radi web aplikacija ..... 5
   3.2. Prednosti web aplikacije ..... 5
4. KORIŠTENE TEHNOLOGIJE KOD IZRADE WEB APLIKACIJE ..... 7
   4.1. HTML ..... 8
   4.2. CSS ..... 10
   4.3. Bootstrap ..... 13
   4.4. JavaScript ..... 13
   4.5. Ajax ..... 15
   4.6. PHP ..... 17
   4.7. SQL ..... 18
      4.7.1. MySQL ..... 19
   4.8. Visual Studio Code ..... 19
5. IZRADA APLIKACIJE ..... 20
   5.1. Baza podataka ..... 20
   5.2. Prijava i registracija korisnika ..... 23
   5.3. Prikaz vanjske temperature ..... 26
   5.4. Preporuke korisniku ..... 28
   5.5. Grafički prikaz vrijednosti ..... 30
   5.6. Prikaz minimalne i maksimalne vrijednosti tijekom dana ..... 32
   5.7. Prikaz svih podataka sa senzora ..... 33
6. KORIŠTENJE APLIKACIJE ..... 36
   6.1. Neregistrirani korisnik ..... 36
   6.2. Registrirani korisnik ..... 38
7. ZAKLJUČAK ..... 44
LITERATURA ..... 45

SAŽETAK.....47
ABSTRACT.....48
ŽIVOTOPIS.....49
PRILOZI.....50

# 1. UVOD

Temperatura i vlaga su jedni od najvažnijih čimbenika koji utječu na prostor u kojem živimo ili boravimo. Nepovoljnim odnosom temperature i vlage dolazi do tri najvažnija utjecaja, a to su nekretnina, zdravlje i ekonomski utjecaj. Zbog pojave plijesni stradavaju materijali i vizualno djeluju neprivlačno i izazivaju neugodne mirise. Pojava bakterija, virusa i gljivica nepovoljno utječu na zdravlje. Jedan od najčešćih problema su razne alergije (kožne i dišne), posebno kod djece. Ekonomski faktor najviše se očituje u troškovima grijanja i gubitkom vrijednosti nekretnine. Zbog eliminacije spomenutih loših utjecaja težnja je praćenje vrijednosti temperature i vlage te upozoravanje na granične vrijednosti i savjetovanje korisnika. Senzori prikupljaju podatke o vrijednostima temperature i vlage. Težnja je postaviti što veći broj senzora kako bi više prostorija bilo praćeno kao i zbog međusobnog utjecaja prostora.

Ovaj završni rad je podijeljen u 5 cjelina. U prvoj cjelini su opisane slične web aplikacije za prikupljanje i prikaz podataka. Potom, slijedi kratak opis i definicija web aplikacije te navedene prednosti korištenja web aplikacije. U trećoj cjelini su detaljnije opisane tehnologije potrebne za uspješnu izradu aplikacije. U četvrtoj su opisane funkcionalnosti aplikacije te princip njene izrade. U petoj, posljednjoj cjelini, su prikazane mogućnosti korištenja aplikacije te neke od provjera prethodno implementiranih funkcionalnosti.

## 1.1. Zadatak završnog rada

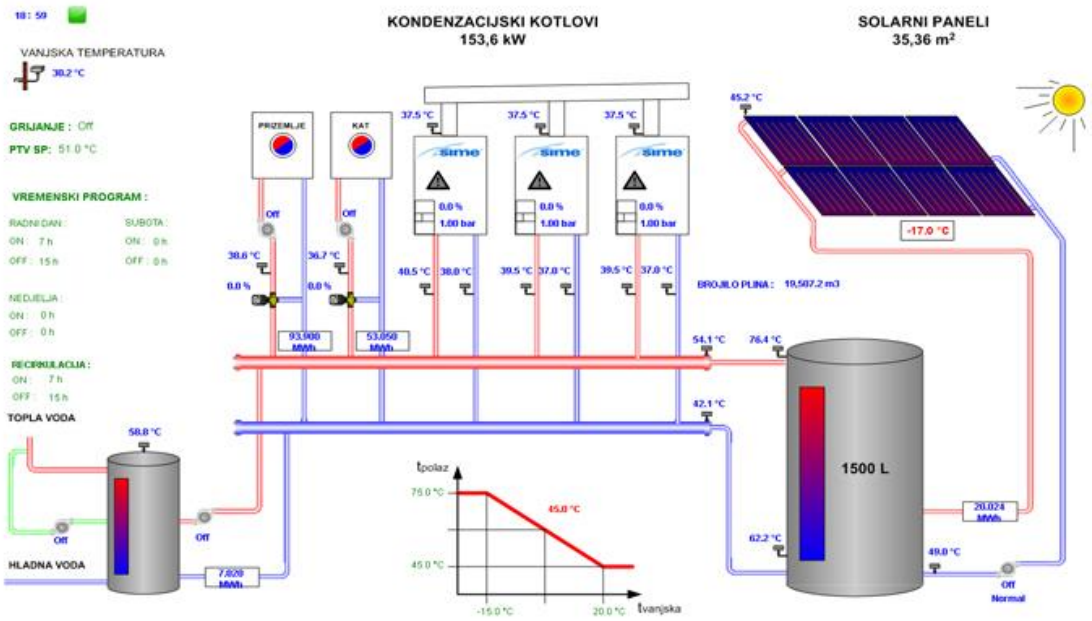
Web aplikacija omogućava korisniku uvid u vrijednosti temperature i vlage pojedinih prostorija unutar kuće. Ovisno o vrijednostima očitanih sa senzora korisnika se informira o toleranciji pojedinih vrijednosti, po potrebi ukazuje mu se što treba učiniti, npr. prozračiti prostoriju, pojačati grijanje i sl. Korisnik ima uvid u 24 satno očitanje s najmanjom i najvećom vrijednosti tijekom dana. Svaka očitana vrijednost prikazana je unutar intervala tolerancije. Omogućen je prikaz vremenske prognoze te topografski prikaz kuće s pozicijom senzora unutar kuće.

# 2. POSTOJEĆE APLIKACIJE

U ovom poglavlju dan je prikaz sličnih aplikacija za prikupljanje i prikaz određenih podataka i automatizaciju procesa.

## 2.1. Cras Telecontrol Service

CTS (engl. *Cras Telecontrol Service*) upravlja i nadzire tehnološke procese poput nadzora staklenika, pumpnih stanica, navodnjavanja kuća, hotela, farmi te se koristi u aplikacijama gdje se automatizacijom određenih procesa povećava učinkovitost sustava. Za pristup aplikaciji za automatizaciju staklenika potrebno se prijaviti. Podaci se prikazuju u stvarnom vremenu te ih se može filtrirati za željeno vremensko razdoblje. Moguće je kontrolirati sustav na daljinu te aplikacija ima podršku i za mobilne uređaje. Slika 2.1. prikazuje izgled CTS-a [1].




**Slika 2.1.** *Izgled Cras Telecontrol Service-a.*

## 2.2. Humidex myHome aplikacija

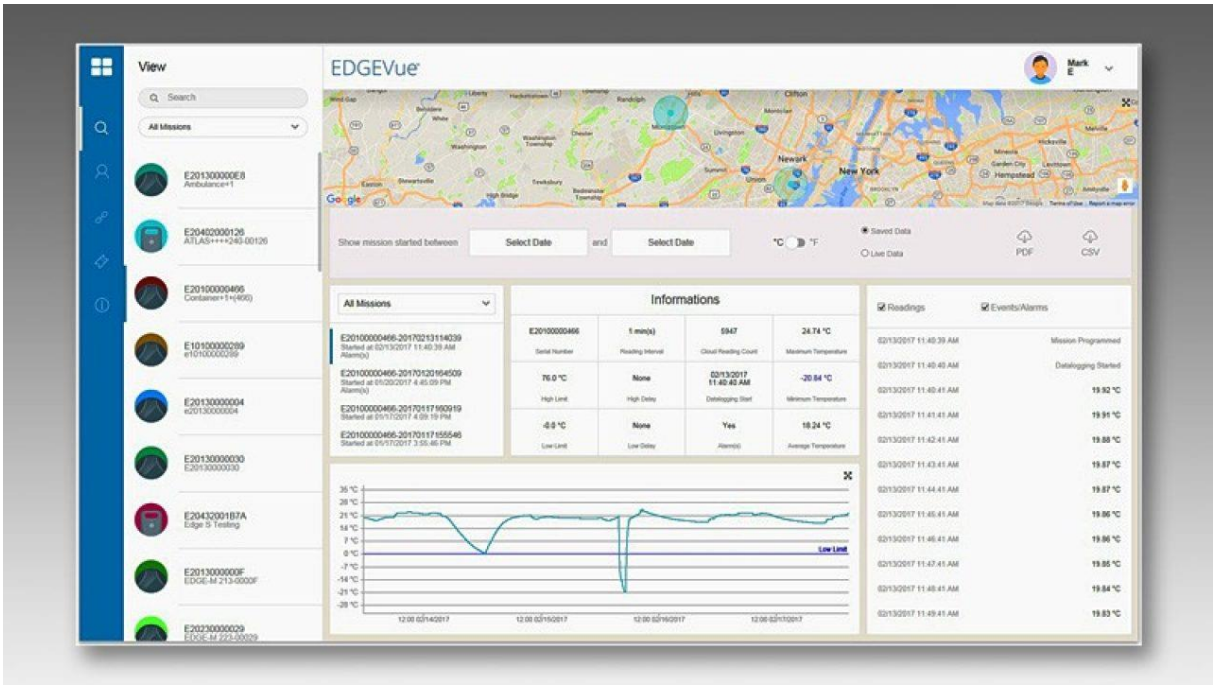
Aplikacija pomoću Humidex sustava precizno kontrolira razinu vlažnosti u kući. Optimizira kvalitetu zraka tijekom godine te je namijenjena za mobilne uređaje i tablete.

Omogućuje smanjenje plijesni i kondenzacije, uklanjane ugljikovog monoksida, neugodnih mirisa, alergena i štetnih tvari te pomaže korisniku da izbjegne štetu na namještaju ili odjeći koju bi uzrokovala plijesan. Pruža i praćenje stanja uređaja u stvarnom vremenu i provjeru vremenske prognoze. Slika 2.2. prikazuje Humidex myHome aplikaciju [2].

**Slika 2.2.** *Humidex myHome aplikacija.*

## 2.3. EDGEVue web aplikacija

EDGEVue web aplikacija omogućuje pristup, pregled i upravljanje podacima prenesenih sa senzora temperature EDGE M i EDGE S-serije. Potrebno se pretplatiti na EDGECloud te se tamo prenose svi podaci kojima korisnik može pristupiti bilo kada i bilo gdje putem web preglednika. Za pristup na „oblak“ (engl. *cloud*) je potrebna prijava. Senzori prate temperaturu osjetljivih proizvoda tijekom transporta. Web aplikacija prikazuje i mjesto na kojem su senzori očitani i sve alarme koji su se mogli pojaviti što je prikazano na slici 2.3. [3].




**Slika 2.3.** *EDGEVue web aplikacija.*

# 3. WEB APLIKACIJA

Web aplikacija je računalni program koji koristi web preglednik i web tehnologiju za obavljanje zadataka putem interneta. Princip povezanosti na kojem radi je klijent-server te se klijentu pruža grafičko sučelje koje je definirano na serveru. Koriste kombinaciju skripti na strani poslužitelja PHP (engl. *Hypertext Preprocessor*) i ASP (engl. *Active Server Pages*) za rukovanje pohranom i dohvaćanjem informacija, a skripte na strani klijenta JavaScript i HTML (engl. *HyperText Markup Language*) kako bi prezentirali informacije korisnicima te njih internet preglednik lako može pročitati i izvoditi operacije koje su programirane. Omogućuju zaposlenicima izradu dokumenata, razmjenu informacija, suradnju na projektima i rad na zajedničkim dokumentima bez obzira na lokaciju ili uređaj. Poznate web aplikacije su: Gmail, Microsoft 365, Google Docs, Google Apps i slične aplikacije tehnoloških giganta.

## 3.1. Kako radi web aplikacija

Web aplikacije se ne moraju preuzimati jer im se pristupa putem mreže. Korisnici mogu pristupiti web aplikaciji putem web preglednika kao što su Google Chrome i Mozilla Firefox. Zahtijeva web poslužitelj za upravljanje zahtjevima od klijenta i aplikaciju za izvršavanje zatraženih zadataka. Ponekad radi i s nekom bazom podataka koja služi za pohranu podataka. Tipičan tijek web aplikacije izgleda ovako:

- Klijent (korisnik) aktivira zahtjev na web poslužitelju putem interneta u web pregledniku.
- Web poslužitelj prosljeđuje zahtjev odgovarajućoj aplikaciji na poslužitelju.
- Aplikacija izvršava traženi zadatak, zatim generira rezultate traženih podataka.
- Aplikacija šalje obrađene podatke web poslužitelju.
- Web poslužitelj prosljeđuje rezultat klijentu u web preglednik.

## 3.2. Prednosti web aplikacije

Web aplikacije rade na više platformi bez obzira na operacijski sustav (MS Windows, Linux, macOS, UNIX) [4].

Jedino što je bitno je imati instaliran preglednik koji podržava izvođenje aplikacije. Svi korisnici pristupaju istoj verziji što uklanja probleme s kompatibilnošću. Ne instaliraju se na tvrdi disk, čime se uklanjaju ograničenja prostora.

Smanjuju se troškovi i za poslovnog i za krajnjeg korisnika zato što se održavanje i nadogradnje web aplikacije izvode na samom serveru.

# 4. KORIŠTENE TEHNOLOGIJE KOD IZRADE WEB APLIKACIJE

Postoji mnogo programskih jezika i tehnologija pomoću kojih se može isprogramirati ili napisati aplikacija. Nema ih potrebe sve nabrajati, no u nastavku ovog poglavlja su navedene i objašnjene one tehnologije koje su izabrane za ovaj rad. Dijele se u dvije skupine, *frontend* i *backend*. Tehnologije za izradu web aplikacije se mogu vidjeti na slici 4.1. [5].

**Slika. 4.1.** *Tehnologije za izradu web aplikacije.*

Frontend tehnologije služe za prikaz vizualnog dijela web aplikacije. Uključuju sve što korisnik izravno doživljava, od teksta i boja do gumba, slika i navigacijskih izbornika. Bitno je napomenuti da, iako se razvoj frontenda bavi vizualnom i interaktivnom stranom web stranice, to nije isto što i web dizajn. Web dizajneri se bave dizajnom, izgledom i dojmom web stranice, dok frontend programer uzima taj dizajn i ugrađuje ga u nešto funkcionalno koristeći jezike kao što su HTML, CSS, JavaScript i drugi [6]. Frontend tehnologije su primjer statičke web stranice, njezin sadržaj se ne mijenja puno. Statičke web stranice su dobre za prikazivanje stvari poput profesionalnih profila, poduzeća, restorana.

Backend tehnologije se odnose na sve što korisnik ne može vidjeti u pregledniku, poput baze podataka i servera [7]. Odgovorne su za pohranu i organiziranje podataka i osiguravanje da sve na strani klijenta ispravno radi.

Backend komunicira s frontendom, šaljući i primajući informacije koje će se prikazati kao web stranica. Kad god klijent ispuni kontakt obrazac, obavi transakciju proizvoda, upiše web adresu ili na bilo koji način obavi interakciju, preglednik šalje zahtjev poslužitelju koji vraća informacije u obliku frontend kôda koji preglednik može interpretirati i prikazati. Backend programeri koriste programske jezike kao što su PHP ili .Net, jer trebaju raditi s nečim što baza podataka razumije.

Backend tehnologije su primjer dinamičke web stranice koja se u stvarnom vremenu stalno mijenja i ažurira. Postoji puno više dinamičkih web stranica u odnosu na statičke. Neke od njih su Facebook, YouTube, Google Maps koje stalno mijenjaju i ažuriraju svoj sadržaj. Dinamička web stranica zahtijeva da baza podataka radi pravilno. Sve se informacije spremaju u bazu podataka, poput korisničkih profila, raznih slika, vrijednosti ili postova na blogu.

## 4.1. HTML

*HyperText Markup Language* ili skraćeno HTML je jezik koji se koristi za izradu web stranica. HTML preko svojih oznaka govori web pregledniku kako rasporediti sadržaj na web stranici i kako ona treba izgledati. Preko njega web stranice komuniciraju s web preglednicima te se može reći da je službeni jezik za komunikaciju preko interneta. HTML nije programski jezik, nego služi za opis hipertekstualnih dokumenata. Ne može vršiti računske operacije, izvoditi petlje i ispitivati uvjete.

Jezik je izumio Tim Berners-Lee. Došao je na ideju internetskog hiperteksta. Hipertekst je tekst koji sadrži linkove ili reference na druge tekstove kojima gledatelji odmah mogu pristupiti. Autor originalnog teksta dodaje reference prema tekstovima drugih autora koji se nalaze na drugim računalima u mreži. Svaki od autora može mijenjati, dopunjavati, poboljšavati i brisati svoj tekst. Time se dobiva globalni hipertekstualni sustav u kojem je gotovo nemoguće ostati bez relevantne informacije.

Prva verzija HTML-a je objavljena 1991. godine, a sastojala se od 18 HTML oznaka. Od tada, svaka nova verzija HTML jezika dolazi s novim oznakama i atributima. Trenutno postoji 140 HTML oznaka, no mnoge su već zastarjele jer ih ne podržavaju moderni preglednici [8].

HTML specifikacije održava i razvija World Wide Web konzorcij (W3C). Oznake i atributi se pišu po pravilu prikazanom na slici 4.2.

```
<oznaka atribut = „vrijednost“>
Tekst za prikaz na stranici.
<!-- Ovo je komentar i neće se prikazati na stranici -->
</oznaka>
```

**Slika 4.2.** *Prikaz oznaka i atributa u HTML-u.*

Oznake i atributi imaju unaprijed definirana imena i svrhu. Na primjer, oznaka koja opisuje font kojim je pisan tekst nosi ime <*font*>, a atribut kojim definiramo veličinu tog fonta nosi naziv *size*.

Svaki HTML dokument treba imati tri oznake, a to su <*html*>, <*head*> i <*body*>.

<*html*> označava početak HTML dokumenta.

<*head*> element predstavlja zaglavlje HTML dokumenta u kojem se nalazi naslov stranice, jezične značajke, dodaju se stilska obilježja stranice i često se definiraju skripte kreirane u JavaScript jeziku.

<*body*> oznaka obuhvaća sav sadržaj koji se nalazi na stranici.

Za oblikovanje teksta unutar HTML dokumenta mora se staviti određena oznaka na početku teksta koji oblikujemo te završna oznaka na kraju teksta što se vidi na slici 4.3.

```
<!DOCTYPE html>
<html>
<head>
<title>Naslov</title>
</head>
<body>

<p>oznaka za odlomak</p>
<big>veliki tekst</big> <br>
<b>podebljani tekst</b> <br>
<u>podcrtan tekst</u> <br>
<strong>"jaki" tekst</strong> <br>
<i>nakrivljen tekst</i> <br>
<del>precrtan tekst</del> <br>

</body>
</html>
```

oznaka za odlomak

veliki tekst
**podebljani tekst**
podcrtan tekst
**"jaki" tekst**
*nakrivljen tekst*
~~precrtan tekst~~

**Slika 4.3.** *Primjer oblikovanja teksta unutar HTML dokumenta.*

Posljednja verzija HTML-a nazvana HTML5 je izašla 2014. godine te je uvela mnoge nove značajke u jezik. Jedna od najbitnijih je izvorna podrška za ugradnju audio i video zapisa umjesto korištenja Flash Playera. Također uključuje ugrađenu podršku za skalabilnu vektorsku grafiku, matematičke i znanstvene formule [9].

Uvela je i nekoliko semantičkih poboljšanja. Nove semantičke oznake preglednicima govore o značenju sadržaja, što koristi i čitateljima i tražilicama.

## 4.2. CSS

CSS (engl. *Cascading Style Sheets*), u prijevodu kaskadni stilovi, jednostavan je mehanizam za oblikovanje web stranica pomoću kojeg možemo dodavati fontove, boje, margine. Nastao je krajem 1994. godine. Prvu specifikaciju CSS-a su objavili Håkon Wium Lee i Bert Bos. Korištenjem CSS-a moguće je odvojiti prikaz podataka i dizajn od strukture podataka, a samim time se dobije pregledniji HTML kôd te ga je lakše kontrolirati.

CSS kôd sastoji se od tri glavna elementa:

1. selektori – identificiraju određene elemente na HTML stranici i utječu na njihov izgled.
2. svojstva – pomoću njih se opisuju pojedina svojstva kao što su boja teksta, vrijednosti margine, poravnanje teksta.
3. vrijednosti – predstavljaju vrijednost koju prima pojedino svojstvo.

Osnovna sintaksa CSS-a je prikazana na slici 4.4.

```
selektor { svojstvo: vrijednost; }
```

**Slika 4.4.** *Osnovna sintaksa CSS-a.*

Komentari se pišu između znakova /* i */.

Postoje četiri moguća načina na koje možemo povezati CSS s HTML-om:

1. Umetanje u zaglavlje dokumenta (unutar oznake *head*).
2. Dodavanje unutar linije HTML kôda pomoću atributa *style.*

3. Povezivanje s vanjskim dokumentom.
4. Umetanje vanjskog dokumenta.

U CSS-u ima nekoliko osnovnih tipova selektora, a to su: jednostavni selektori, klasni selektori, id selektori i pseudoklase. Jednostavni selektori su najjednostavniji tip selektora i imaju imena jednaka html oznakama te zbog toga ne zahtijevaju izmjene HTML kôda. Problem nastaje ako selektori imaju isto ime, tako da će se selektor p odnositi na sve odlomke unutar web stranice.

Ako želimo da različiti odlomci ne izgledaju isto morat ćemo koristiti druge selektore kao što su klasni selektori. Klasni selektor se najčešće odnosi na više elemenata. Definira se tako da se ispred imena selektora stavi točka. Id selektor se koristi za definiranje samo jednog HTML elementa na web stranici. Definira se tako da se ispred imena selektora stavi oznaka #.

Prema slici 4.5. univerzalni je selektor označen znakom zvjezdica (*). On ima najveći prioritet te definira da će svi elementi biti pisani fontom *Arial*. Nakon njega je klasni selektor naslov koji se odnosi na sve klase koje imaju taj naziv. Određuje da će naslov teksta biti centriran i plave boje. Posljednji je id selektor koji se odnosi na samo taj odlomak te je jedinstven na web stranici. Određuje da će margine za odlomak na sve četiri strane iznositi 10 piksela te da će veličina fonta biti 16 piksela.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<h1 class="naslov">Naslov teksta</h1>
<p id="odlomak1">Odlomak u koji upisujemo tekst.</p>

</body>

<style>
/* Početak CSS-a */
* {
    font-family: arial;
}
.naslov {
  text-align: center;
  color: blue;
}

#odlomak1 {
    margin: 10px;
    font-size:16px;
    }
/* Kraj CSS-a */
</style>
</html>
```

**Slika 4.5.** *Prikaz označavanja CSS selektora.*

Slika 4.6. prikazuje sintaksu pseudoklase koja je malo drugačija u odnosu na ostale selektore.

**Slika 4.6.** *Sintaksa pseudoklase.*

Najčešće korištene su:

- *link* – izgled neposjećenog linka
- *visited* – izgled posjećenog linka
- *hover* – izgled linka kada se pokazivač miša postavi iznad linka
- *active* – izgled aktivnog linka
- *first-child* – izgled elementa koji je prvi potomak nekom drugom elementu.

Elementi u CSS-u mog naslijediti neka svojstva drugog elementa ako se nalaze unutar njega.

Prema slici 4.7. paragraf ili odlomak nasljeđuje crveni obrub veličine jednog piksela i automatski nasljeđuje *Sans-serif* font.

```
<div style="font-family:serif; border:1px solid red; padding:10px;">
  Ovaj tekst će imati serif font.
<p style="border:inherit;">
  Sada će paragraf također imati crveni obrub. Za to je zaslužna vrijednost „inherit“,
  što znači naslijedi.</p>
</div>
```

**Slika 4.7.** *Nasljeđivanje svojstava elementa.*

Trenutno najnovija verzija CSS-a je CSS3. Glavna razlika je što CSS3 ima module i podržava responzivni dizajn [10]. Omogućava sve vrste prijelaza, animacija i transformacija elemenata. Može dati bilo kojem elementu zaobljene kutove koristeći *border-radius*. Omogućava više pozadinskih slika za jedan element pomoću pozadinskih svojstava. Također ima više svojstava za boje te uvodi svojstvo prozirnosti i transparentnosti.

CSS3 gradijenti mogu prikazivati glatke prijelaze između dvije ili više određenih boja. Definirane su dvije vrsta gradijenata, a to su linearni i radijalni ili kružni gradijenti. Može dodati sjenu na tekst i element.

## 4.3. Bootstrap

*Bootstrap* je najpopularniji besplatni okvir za razvoj responzivnih projekata. Omogućava programerima i dizajnerima bržu izradu responzivnih web stranica. Sadrži HTML, CSS i JavaScript kôd za različite komponente na stranici kao što su slike, skočni prozori, forme, navigacijske trake i ostale komponente. *Bootstrap* dokumentacija je odlično napisana na službenoj web stranici [11]. Svaki dio kôda je detaljno objašnjen i opisan te je prilagođen i početnicima koji mogu razne komponente kopirati i zalijepiti na svoju stranicu i potom prilagoditi dizajnu stranice. Trenutna inačica *Bootstrapa* je *Bootstrap* 5.0.

## 4.4. JavaScript

JavaScript je objektno orijentirani skriptni jezik koji se izvodi na klijentskom računalu. Podržavaju ga svi noviji web preglednici. Sastoji se od linija kôda koji se odmah, liniju po liniju izvršavaju, bez prevođenja cijelog kôda i kreiranja izvršne datoteke. Nije pravi programski jezik jer se programski jezici prije izvođenja moraju prevesti ili kompajlirati u strojni kôd. Izmislio ga je Brendan Eich 1996. godine za Netscape Navigator 2.0.

JavaScript se najčešće koristi za:

- Dodavanje interaktivnosti na web stranicu čime se dobiva dinamička web lokacija.
- Razvoj mobilnih aplikacija.
- Izradu igara temeljenih na web pregledniku.
- Provjeru ispravnosti i vjerodostojnosti podataka prije nego se pošalju na server.
- Pohranu i učitavanje informacija o korisnikovom računalu [12].

Dijeli varijable na cjelobrojne, decimalne, znakovne i logičke varijable. Deklaracija varijabli se vrši izrazom: *var imeVarijable;*

Postoje tri načina na koja se može spremati JavaScript kôd:

1. Spremanje unutar samog HTML dokumenta, tj. unutar oznaka <*head*> ili <*body*>.
2. Upisivanje u polje za adresu.
3. Spremanje u posebnu JavaScript datoteku s ekstenzijom (.js).

Prema slici 4.8. atribut *type* se koristi radi preciznosti i naznake jezika koji slijedi.

```
<script type="text/javascript" src="datoteka.js"></script>
```

**Slika 4.8.** *Primjer poveznice na datoteka.js datoteku.*

Ima nekoliko vrsta operatora koji vrše operacije nad varijablama te mogu mijenjati i vrijednosti varijabli. Naslijedio je strukturno programiranje kao što su *if*, *if-else*, *switch* naredbe, te petlje *for*, *while* i *do-while* što se može vidjeti na slici 4.9.

```
<!DOCTYPE html>
<html>
<body>

<h3>JavaScript For petlja</h3>

<p id="x"></p>

<script>
var cars = ["BMW", "Nissan", "Renault", "Ford", "Fiat", "Audi"];
var text = "";
var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
document.getElementById("x").innerHTML = text;
</script>

</body>
</html>
```

**JavaScript For petlja**

BMW
Nissan
Renault
Ford
Fiat
Audi

**Slika 4.9.** *Primjer korištenja for petlje u JavaScriptu.*

JavaScript ima okidače ili pokretače događaja koji pokreću određene funkcije. Koriste se za izradu interaktivnih dijelova web stranice kao što su gumbi koji reagiraju na klik miša, izbornici, izmjenjivi sadržaji i ostalo.

U njima su definirane metode koje reagiraju na određene akcije korisnika. Pomoću njih znamo kada je posjetitelj kliknuo na link, gumb formulara, prešao pokazivačem preko određenog polja.

Prema slici 4.10. klikom lijevog klika mišem na dugme prikazuje se trenutno vrijeme.

```
<!DOCTYPE html>
<html>
<body>

<p>Klikom na dugme će vam se prikazati trenutni datum.</p>

<button onclick="displayDate()">Vrijeme je?</button>

<script>
function displayDate() {
  document.getElementById("x").innerHTML = Date();
}
</script>

<p id="x"></p>

</body>
</html>
```



**Slika 4.10.** *onClick() događaj.*

JavaScript omogućava definiranje vlastitih objekata i njihovih svojstava. Objekti su specijalni tipovi podataka koji imaju svoja svojstva i metode i mogu imati podobjekte koji nasljeđuju ta svojstva i metode, ali dodaju i nova vlastita svojstva i metode koje su specifične samo za njih.

## 4.5. Ajax

Ajax je kratica za asinkroni JavaScript i XML (engl. *Extensible Markup Language)*. Nije programski jezik, nego predstavlja grupu tehnologija. Koristi se često u programiranju na strani klijenta kako bi se omogućilo slanje i primanje podataka u bazu i iz nje. Web aplikacija koja koristi Ajax tehnologiju se učitava samo jednom dok se komunikacija sa serverom odvija asinkrono. Podaci se razmjenjuju u pozadini, a pritom ne ometaju korisničko iskustvo. Ajax omogućuje web stranicama dinamičku promjenu sadržaja bez potrebe za ponovnim učitavanjem cijele stranice čime se poboljšavaju performanse stranice. Podaci sa servera se mogu tražiti preko konvencionalne metode ili preko Ajax metode.

Prema konvencionalnoj metodi preglednik traži podatke od web poslužitelja te se događa sljedeće:

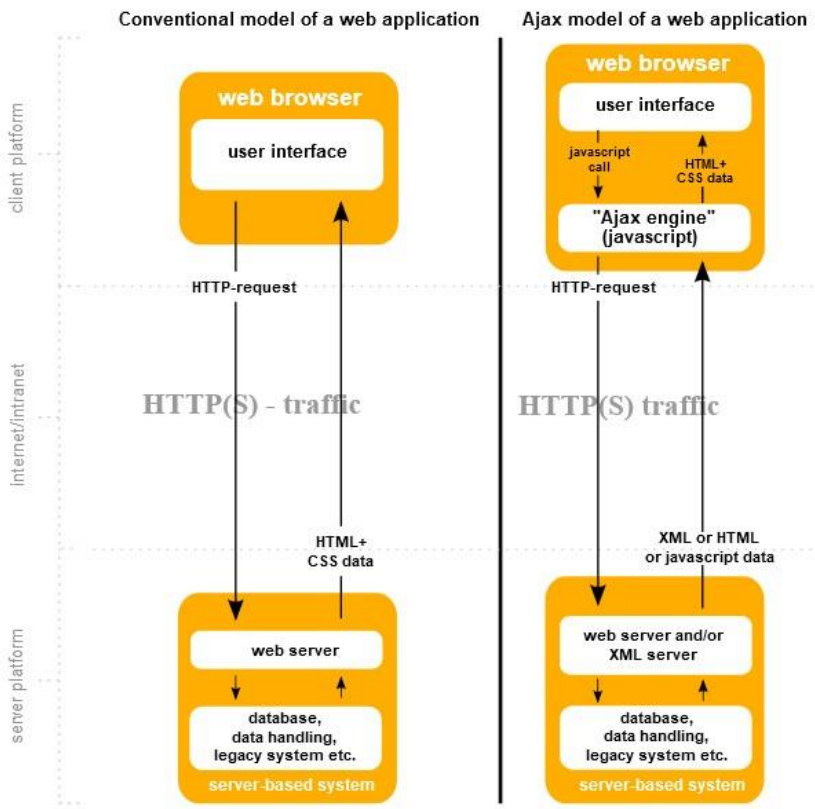
1. HTTP zahtjev se upućuje iz preglednika na web poslužitelj. Korisnik mora pričekati da se zahtjev obradi i vratiti odgovor prije nego što vidi tražene podatke.
2. Zahtjev dolazi do web poslužitelja te se dohvaćaju odgovarajući podaci.

3. Traženi podaci se vraćaju u preglednik i korisnik ih može vidjeti.

Prema Ajax metodi koristeći iste podatke događa se sljedeće:

1. Preglednik izvodi JavaScript poziv prema Ajaxovom mehanizmu, tj. stvara se *XMLHttpRequest* objekt. Njega JavaScript koristi za komunikaciju sa serverom.
2. U pozadini HTTP zahtjev se upućuje poslužitelju i dohvaćaju se odgovarajući podaci.
3. HTML, XML i JavaScript podaci se vraćaju Ajaxovom mehanizmu koji zatim šalje tražene podatke u preglednik.

Korištenjem Ajax metode korisnik ne doživljava zastoje od trenutka kada je podnio zahtjev do trenutka kada je primio stvarne informacije. Slika 4.11. prikazuje usporedbu između konvencionalne metode i Ajax metode traženja podataka s web servera [13].




**Slika 4.11.** *Usporedba konvencionalne metode i Ajax metode traženja podataka.*

Tradicionalno su se skripte učitavale sinkronom metodom. Na taj način se izvršavaju sve skripte prilikom prikaza web stranice što dovodi do sporijeg učitavanja stranice.

Svaka nadolazeća skripta ovisi o prethodnoj pošto mora čekati da prethodna skripta završi s učitavanjem. Asinkrona metoda je učinkovitija pošto postupak učitavanja skripti neće prekinuti prikazivanje web stranice. To također omogućuje istovremeno učitavanje više skripti te se mogu zanemariti one skripte koje su irelevantne za učitavanje početne web stranice. Asinkrona metoda nije dobra opcija ako se skripte trebaju učitati određenim redoslijedom, u tom slučaju je bolja sinkrona metoda.

## 4.6. PHP

PHP je popularan skriptni jezik otvorenog kôda opće namjene, koji je posebno prikladan za web razvoj i može se ugraditi u HTML. Napisao ga je 1995. godine programer Rasmus Lerdorf za održavanje njegove privatne web stranice, odakle i skraćenica „Osobna kućna stranica".

PHP na serveru najprije kreira HTML stranicu i onda je šalje klijentu te tako dinamičko generira kôd. Drugačiji je od JavaScript-a zato što se JavaScript izvršava lokalno na računalu korisnika te server nikad ne pokreće JavaScript kôd.

PHP podupire rad s većinom baza podataka kao što su MySQL, Oracle, PostgreSQL i druge. Također je kompatibilan s većinom servera (Apache, IIS). Sve skripte koje se naprave se mogu testirati lokalno putem programa XAMPP jer on simulira rad servera na korisničkom računalu. Ako server podržava PHP onda nema potrebe instalirati PHP na računalu. PHP kôd se piše između znakova *<?php* i *?>* . To je vrlo korisno jer se tako može razlikovati PHP od HTML kôda. Varijable se pišu tako da se ispred njihovog imena dodaje znak *$* te je na kraju kôda potrebno staviti znak „točka-zarez" što označava da je tvrdnja gotova.

Prema slici 4.12. varijabli *c* dodajemo umnožak brojeva koju sadrži varijabla a i varijabla b. Koristeći *if* naredbu provjeravamo je li umnožak brojeva 20. Tvrdnja je istinita te se preko *echo* naredbe ispisuje na zaslon. Znakovni operator točka se koristi da se spoji više *stringova,* tj. niza znakova.

```php
<!DOCTYPE html>
<html>
<body>

<?php
$a = 5;
$b = 4;
$c = $a * $b;
if ($c == 20) {
        echo "Umnožak je: " .$c;
    }
    else {
        echo "Umnožak je različit od 20";
    }
?>

</body>
</html>
```

Umnožak je: 20

**Slika 4.12.** *Umnožak dva broja u PHP s if naredbom.*

PHP ima brojne mogućnosti, no u nastavku će biti nabrojane samo neke od njih. Može prikupljati podatke obrasca, generirati dinamički sadržaj stranice i slati i primati kolačiće (engl. *cookies*). Kolačići pamte što korisnik radi na web stranici, gdje klika te može zapamtiti čak i lozinke tijekom prijave ako korisnik to omogući. Pamte i što posjetitelj dodaje prilikom kupnje na nekoj web trgovini te se većinom koriste za digitalni marketing.

Prilikom pisanja PHP kôda korisnik ima mogućnost korištenja proceduralnog programiranja ili objektno orijentiranog programiranja ili njihove kombinacije. PHP može automatski generirati XHTML (engl. *Extensible HyperText Markup Language*) ili bilo koju drugu XML datoteku i spremiti ih u datotečni sustav, umjesto da ih ispisuje [14]. XHTML je nadogradnja HTML-a te je osmišljen s ciljem da bude kompatibilan s više tipova podataka te da ima više mogućnosti. Jedna od najznačajnijih značajki PHP-a je podrška širokom rasponu baza podataka. Putem PHP-a korisnik može dodati, brisati ili mijenjati elemente u bazi podataka te može i šifrirati podatke.

## 4.7. SQL

SQL (engl. *Structured Query Language*) je standardni jezik za pristup i upravljanje relacijskim bazama podataka. Definira objekte baze podataka, administraciju korisnika, prava pristupa podacima te relacijsku shemu baze podataka (EER dijagrami i sl.).

Pomoću naredbi *INSERT*, *UPDATE* i *DELETE* korisnik može dodati, modificirati ili obrisati podatke. SQL je nastao početkom 1970-ih te ga je razvila tvrtka IBM.

Osnovni element koji se pohranjuje u bazi je entitet. Entitet može biti objekt, biće, događaj. Svojstvo koje opisuje entitet naziva se atribut. Sljedeći bitan pojam je relacija. Relacije se uspostavljaju između dva ili više entiteta te definiraju njihov međusobni odnos. Mogu biti jedan prema jedan, jedan prema više te više prema više. Baza podataka koristi tablice za pohranu podataka. Tablicu čine stupci i redovi. Vrijednosti pojedinih atributa promatranog podatka se nalaze u stupcima, dok se vrijednosti svih atributa promatranog podatka nalaze u redovima.

Postoje i nerelacijske baze podataka ili NoSQL. One za razliku od SQL-a ne spremaju podatke putem tabličnih relacija nego su orijentirane na dokumente. U takve se baze mogu dodati novi podaci bez da se unaprijed definiraju u shemi baze podataka što omogućuje obradu velike količine nestrukturiranih i strukturiranih podataka [15].

### 4.7.1. MySQL

MySQL je sustav za upravljanje bazama podataka zasnovan na SQL jeziku. To je najpopularniji sustav otvorenog kôda na svijetu [16].

Kao i kod ostalih relacijskih baza podataka, MySQL podatke pohranjuje u tablice sastavljene od redaka i stupaca. Prilagođen je za rad na raznim operacijskim sustavima kao što su Windows, Linux i UNIX. Prilično je brz, optimalno koristi memoriju te ima malu potrošnju resursa. Osigurava optimalnu brzinu te je dizajniran da udovolji najzahtjevnijim aplikacijama kao što su web trgovine koje svaki dan primaju i po milijun upita.

## 4.8. Visual Studio Code

Visual Studio Code je uređivač kôda napravljen od tvrtke Microsoft. Dostupan je za Windows, Linux i MacOS operacijski sustav. Korisnici mogu podesiti prečace na tipkovnici za određene funkcionalnosti te instalirati razna proširenja za dodatne funkcije kao što su automatsko poravnavanje kôda i kreiranje lokalnog servera koji se automatski učitava za statičke i dinamičke web stranice. Kompatibilan je s različitim programskim jezicima. Prema anketi za programere iz 2019. godine Visual Studio Code zauzima 1. mjesto među najpopularnijim razvojnim alatima [17].

# 5. IZRADA APLIKACIJE

U ovom poglavlju bit će opisani kôd koji je napisan korištenjem backend i frontend tehnologija. Na početku su napravljene tri PHP datoteke naziva *header.php, index.php* i *footer.php*. *Header.php* datoteka je najbitnija pošto se u njoj nalazi *html* oznaka koja označava početak HTML stranice te se nalaze definirane skripte za *Bootstrap 4*, kalendarski prikaz, Google grafikone, Ajax te vanjska CSS datoteka u kojoj se nalazi sav CSS kôd koji opisuje kako HTML oznake trebaju izgledati. Njih je obvezno umetnuti zato što se inače ne bi mogle koristiti te tehnologije i neke funkcionalnosti koje su bile potrebne za izradu ove web stranice. Tamo se još nalazi navigacijska traka koji ima dugme za prijavu i registraciju korisnika.

*Index.php* datoteka predstavlja sredinu stranice gdje je većina kôda dok *footer.php* datoteka je podnožje stranice unutar koje su veze na vanjske skripte napravljene u JavaScript jeziku. Slika 5.1. prikazuje povezivanje *index.php* datoteke s *header.php* i *footer.php* datotekom. U ovom slučaju se koristi *require* naredba koja u slučaju neuspjeha javlja grešku i zaustavlja skriptu, a da se koristi *include* naredba ona bi samo javila upozorenje i skripta bi nastavila raditi.

index.php

```php
<?php
    require "header.php";
?>

<div>

</div>


<?php
    require "footer.php";
?>
```

**Slika 5.1.** *Povezivanje index.php datoteke s header.php i footer.php datotekom.*

## 5.1. Baza podataka

Softverski alat koji je korišten za kreiranje MySQL baze podataka je *phpMyAdmin*. Besplatan je za korištenje te je napisan u PHP skriptnom jeziku. Ima brojne mogućnosti poput stvaranja, mijenjanja, dohvaćanja, brisanja tablica ili podataka koji su u tablici.

Korisnik može uvoziti neku drugu bazu podataka ili tablicu koja je prethodno stvorena i isto tako može i izvesti te podatke. Također može provjeravati, uspoređivati i optimizirati tablice te izvršavati i brojne druge naredbe nad podacima. Sadrži i korisničko sučelje gdje se većina tih operacija može jednostavno implementirati, ali korisnik ima mogućnost izravnog upisivanja SQL upita što je bolja praksa za samo razumijevanje jezika.

Za ovaj rad osmišljena je baza podataka naziva *webapp* u kojoj se nalaze tri tablice. *Users* tablica prikazuje podatke o korisnicima koji su prijavljeni na bazu, *table_data* tablica sadrži podatke o temperaturi, vlazi, datumu i vremenu te senzorima koji se nalaze u prostorijama unutar kuće.

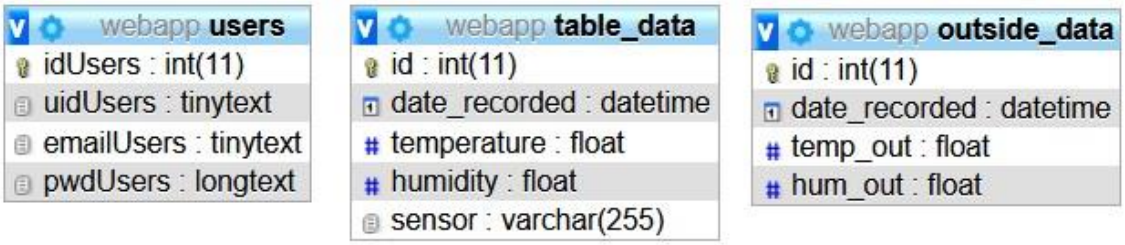
*Outside_data* tablica prikazuje iste podatke kao i *table_data* tablica samo što nema stupac za senzore te sadrži prosječnu vanjsku temperaturu i vlažnost svakoga dana unesenih s „Weather underground“ web stranice [18]. Navedene podatke je moguće vidjeti na slici 5.2.

| ←T→ | | | id | date_recorded | temp_out | hum_out |
|---|---|---|---|---|---|---|
| ☐ | Edit | Copy Delete | 1 | 2020-08-28 00:00:00 | 22.3 | 70.1 |
| ☐ | Edit | Copy Delete | 2 | 2020-09-15 00:00:00 | 21.6 | 71.5 |
| ☐ | Edit | Copy Delete | 3 | 2021-02-23 00:00:00 | 10.2 | 65.8 |
| ☐ | Edit | Copy Delete | 4 | 2021-03-07 00:00:00 | 0.7 | 65 |
| ☐ | Edit | Copy Delete | 5 | 2021-03-16 00:00:00 | 5.8 | 61.8 |
| ☐ | Edit | Copy Delete | 6 | 2021-04-19 00:00:00 | 7.9 | 84.7 |
| ☐ | Edit | Copy Delete | 7 | 2021-04-20 00:00:00 | 9.6 | 82.7 |
| ☐ | Edit | Copy Delete | 8 | 2021-04-22 00:00:00 | 13 | 62.3 |
| ☐ | Edit | Copy Delete | 9 | 2021-04-23 00:00:00 | 12.3 | 54 |
| ☐ | Edit | Copy Delete | 10 | 2021-04-24 00:00:00 | 11.2 | 59 |
| ☐ | Edit | Copy Delete | 11 | 2021-04-25 00:00:00 | 12.8 | 57.4 |
| ☐ | Edit | Copy Delete | 12 | 2021-04-26 00:00:00 | 10.6 | 59.5 |
| ☐ | Edit | Copy Delete | 13 | 2021-04-27 00:00:00 | 7.9 | 90.3 |
| ☐ | Edit | Copy Delete | 14 | 2021-04-28 00:00:00 | 9.8 | 87.7 |

**Slika 5.2.** *Prosječne vrijednosti temperature i vlažnosti po danima.*

Iako je korištena relacijska tablica, tablice nisu povezane na uobičajen način gdje se primarni ključ jedne tablice povezuje sa stranim ključem druge tablice. To nije bilo potrebno za ovu stranicu jer su se podaci između tablica mogli povezati i preko SQL upita.

Na slici 5.3. se vidi prikaz svih kreiranih tablica pomoću opcije *Designer* u *phpMyAdmin* alatu. *Designer* opcija također grafički može prikazati relacije između tablica ako ih ima. Primarni ključ za sve tablice je *id* koji je obvezan za svaku tablicu, mora biti jedinstven za svaki podatak i ne smije poprimiti *NULL* vrijednost. Maksimalno jedan primarni ključ može biti u jednoj tablici.




**Slika 5.3.** *Prikaz baze podataka.*

Slika 5.4. prikazuje jednostavan SQL upit za kreiranje tablice *table_data*. Unosom svakog novog podatka *id* se automatski povećava za jedan preko *auto_increment* naredbe.

```
1  CREATE TABLE table_data (
2      id INT(11) PRIMARY KEY AUTO_INCREMENT NOT NULL,
3      date_recorded DATETIME NOT NULL,
4      temperature FLOAT NOT NULL,
5      humidity FLOAT NOT NULL,
6      sensor VARCHAR(255) NOT NULL
7      );
```

**Slika 5.4.** *SQL upit za kreiranje tablice table_data.*

Spajanje na lokalnu bazu podataka se odvija preko XAMPP-a koji će biti detaljnije objašnjen u poglavlju gdje se opisuje korištenje web stranice. U PHP datotekama se koristi *MySQLi* proširenje baze podataka, a ne *MySQL*. To je poboljšana verzija MySQL-a koje nudi proceduralan i objektno orijentirani način programiranja te podržava pripremljene izjave (engl. *prepared statements*) što prije nije bilo moguće korištenjem MySQL proširenja.

Spajanje na lokalnu bazu je prikazano na slici 5.5. Varijabla *connect* sadrži sve parametre koji su joj potrebni za uspješno spajanje. U slučaju spajanja na online bazu podataka potrebo je na stranici koja pruža hosting otići na upravljačku ploču te pronaći naziv servera i ubaciti ga u varijablu *serverName*. Nadalje, gdje god je potrebno spojiti se na bazu mora se *„dbh.php“* datoteka staviti u *require* naredbu unutar PHP kôda.

registration > dbh.php

```php
<?php

$serverName = "localhost";
$dbUsername = "root";
$dbPassword = "";
$dbName = "webapp";

$connect = mysqli_connect($serverName, $dbUsername, $dbPassword, $dbName);

if (!$connect) {
    die("Connection failed: ".mysqli_connect_error());
}
```

**Slika 5.5.** *Prikaz datoteke „dbh.php“ koja služi za spoj na bazu podataka.*

## 5.2. Prijava i registracija korisnika

Registracija novog korisnika i prijava je potrebna kako bi se vidjeli svi podaci na web stranici. HTML forma omogućuje korisniku unos podataka koji se šalju serveru na obradu. „Registration.php“ datoteka sadrži formu za registraciju i prijavu. Forma za registraciju sadrži polje za unos korisničkog imena, e-mail adrese, lozinke te ponovnog unosa lozinke čime se povećava sigurnost. Forma poziva PHP skriptu *registration/signup.php* u kojoj se rade razne provjere podataka. Forma za prijavu sadrži polje za unos korisničkog imena i lozinke te poziva PHP skriptu *registration/login.php* koja je vrlo slična signup skripti. Obje forme kao način slanja podataka koriste POST metodu. Pri POST metodi korisnik ne vidi što forma šalje skripti te je zbog toga sigurna za upotrebu. Postoji i GET metoda u kojoj su svi podaci vidljivi u URL traci web preglednika.

Slika 5.6. prikazuje neke od provjera prilikom registracije novog korisnika. Definirane su četiri varijable koje prikupljaju podatke iz HTML obrasca preko imena oznake polja za unos. Provjerava se da li su sva polja popunjena, ispravnost unesene e-mail adrese, poklapanje lozinki.

Funkcija *preg_match* provjerava da li je korisničko ime unutar znakova a – z i znamenki 0 - 9 te vraća 1 ako je tvrdnja istinita. Analogno tome, na sličan način su napisane provjere za prijavu postojećeg korisnika. Ako je provjera neuspješna korisnika se šalje na web lokaciju unutar *header* funkcije.

```php
<?php
if (isset($_POST['signup-submit'])) {

    require 'dbh.php';

    $username = $_POST['uid'];
    $email = $_POST['mail'];
    $password = $_POST['pwd'];
    $passwordRepeat = $_POST['pwd-repeat'];

    if (empty($username) || empty($email) || empty($password) || empty($passwordRepeat)) {
        header("Location: ../registration.php?error=emptyfields&uid=".$username."mail=".$email);
        exit();
    }

    else if (!filter_var($email, FILTER_VALIDATE_EMAIL) && !preg_match("/^[a-zA-z0-9]*$/", $username)) {
        header("Location: ../registration.php?error=invalidmailuid");
        exit();
    }

    else if ($password !== $passwordRepeat) {
        header("Location: ../registration.php?error=passwordcheck&uid=".$username."mail=".$email);
        exit();
    }
```

**Slika 5.6.** *Provjere prilikom registracije novog korisnika.*

Pripremljene izjave su vrlo bitne za sigurnost korisničkih podataka i baze. Potrebno ih je pisati jer u protivnom napadač može umetnuti SQL kôd u formu za unos korisničkih podataka i tako prouzročiti *SQL injection.* Na taj način može doći do osjetljivih podataka o korisniku ili može onemogućiti ili srušiti cijelu bazu podataka te imati ovlasti nad brojnim drugim mogućnostima koje su nepovoljne za korisnika.

Kôd koji provjerava da li je korisničko ime zauzeto u bazi koristeći pripremljene izjave je na slici 5.7. Parametar „?“ zamjenjuje korisničko ime u ovom slučaju. Funkcija *mysqli_stmt_init* inicijalizira izraz te kao parametar prima *connect* varijablu za spoj na bazu podataka. Nakon toga se priprema izjava za izvršenje. Funkcija *mysqli_stmt_bind_param* veže varijable kao parametre te parametar „s“ označava argument koji predstavlja niz znakova (engl. *string*).

Taj argument još može biti cjelobrojni, realni ili binarni tip podatka, ali u ovom slučaju mora biti niz znakova jer se odnosi na korisničko ime. Nakon toga se izjava izvrši te se podaci iz tablice dohvaćaju i spremaju u varijablu *resultCheck.*

```
$sql = "SELECT uidUsers FROM users WHERE uidUsers=?";
$stmt = mysqli_stmt_init($connect);

if (!mysqli_stmt_prepare($stmt, $sql)) {
header("Location: ../registration.php?error=sqlerror");
exit();
}

else {
    mysqli_stmt_bind_param($stmt, "s", $username);
    mysqli_stmt_execute($stmt);
    mysqli_stmt_store_result($stmt);
    $resultCheck = mysqli_stmt_num_rows($stmt);
    if ($resultCheck > 0) {
        header("Location: ../registration.php?error=usertaken&mail=".$email);
        exit();
    }
```

**Slika 5.7.** *Provjera da li je korisničko ime zauzeto korištenjem pripremljenih izjava.*

Još jedna bitna stavka kod sigurnosti je raspršivanje lozinke (engl. *password hash*). Funkcija raspršivanja uzima ulaznu vrijednost, tj. lozinku unesenu u formu za prijavu i pretvara ju u dugački niz slova i znakova fiksne veličine. Zbog toga je lozinka u tablici *users* znakovnog tipa *longtext*. Tako admin ne može vidjeti lozinke korisnika u bazi. Isto tako, napadač ako ima pristup bazi neće moći dešifrirati raspršenu lozinku.

Kada se postojeći korisnik ponovno pokuša prijaviti na stranicu i unese lozinku, algoritam raspršivanja provjerava odgovara li trenutno unesena raspršena lozinka postojećoj raspršenoj lozinki u bazi podataka. Ako se poklapaju, korisnik će biti prijavljen na stranicu. Funkcija koja je korištena za raspršivanje lozinke je *password_hash* koja prima dva parametra. Prvi je lozinka iz baze podataka, a drugi algoritam *bcrypt* naziva *PASSWORD_DEFAULT*. *Bcrypt* algoritam se konstantno ažurira te je prilično siguran i pouzdan algoritam za raspršivanje lozinke u odnosu na druge algoritme.

Nakon uspješne prijave korisnika na stranicu bilo je potrebno stvoriti mehanizam zvan sesije. U ovom radu varijable sesije provjeravaju da li je korisnik prijavljen na stranicu. Pohranjuju korisnikov id i ime, ali mogu pohraniti i druge informacije ovisno o vrsti web stranice što olakšava rad korisniku koji ih upotrebljava. Sesije traju sve dok korisnik ne napusti stranicu te se ponovno pozivaju iste varijable kada se korisnik vrati na stranicu. Sesija se započinje funkcijom *session_start* koja se nalazi prije HTML oznake u *header.php* datoteci. Kada se korisnik odjavljuje sa stranice poziva se funkcija *session_unset* koja uklanja varijable sesije, tj. korisnikov id i ime te se poziva *session_destroy* funkcija koja uništava sesiju. Te funkcije se nalaze u *logout.php* datoteci.

Slika 5.8. prikazuje kako se sadržaj mijenja ovisno da li je korisnik prijavljen ili ne. Ako nije prijavljen onda vidi dugme za prijavu, a ako je onda se to dugme neće prikazati nego će se pojaviti dugme za odjavu.

```php
<?php
    if(isset($_SESSION['userId'])) {
        echo    '<form action="registration/logout.php" method="post">
                <li class="nav-item">
                    <button type="submit" class="logout" name="logout-submit">Logout</button>
                </li>
                </form> ';
    }

    else {
        echo    '<form action="registration.php" method="post">
                <li class="nav-item">
                    <button type="submit" class="login_signup">Login/Signup</button>
                </li>
                </form> ';
    }
?>
```

**Slika 5.8.** *Mijenjanje sadržaja na stranici ovisno da li je korisnik prijavljen ili ne.*

## 5.3. Prikaz vanjske temperature

Neregistrirani korisnik može vidjeti vanjsku vrijednost temperature ovisno o lokaciji gdje se nalazi. Za dohvaćanje trenutne vremenske prognoze korištena je „*OpenWeatherMap“* web stranica. Na stranici je potrebno generirati *API* ključ koji se dodaje u JavaScript datoteku. *API* ključ se koristi za identifikaciju ili provjeru autentičnosti web aplikacije ili korisnika.

Nije toliko siguran kao token koji služi za potrebe autentifikacije, no pomaže pri identificiranju aplikacije koja stoji iza toga poziva.

Funkcionalnost za dohvaćanje temperature izvana i trenutne lokacije napisana je unutar *weather.js* JavaScript datoteke. Za pristup HTML elementu, deklarirane su varijable koje koriste *document.getElementById(id)* metodu gdje atribut *id* definira HTML element. U varijable *lat* i *long* se spremaju vrijednosti zemljopisne širine i dužine. *Navigator.geolocation* svojstvo vraća objekt koji web sadržaju daje pristup lokaciji uređaja. Prije nego što uređaj dobije podatke o trenutnoj lokaciji, korisnika se obavijesti i zatraži dopuštenje u web pregledniku.

Uz *API* ključ potrebno je dodati i *proxy* link zbog sigurnosti podataka i pravila istog podrijetla (engl. *same-origin policy).* Pravilo istog podrijetla je sigurnosni mehanizam koji ograničava način na koji dokument iz jednog izvora stupa u interakciju s resursom iz drugog izvora. Pomaže izolirati potencijalno zlonamjerne dokumente, smanjujući moguće napade. Dvije stranice imaju isto podrijetlo ako im je protokol, broj porta i ime hosta jednak.

Na slici 5.9. je dio kôda za *weather.js* datoteku. Podaci koji se dohvaćaju su prema zadanim postavkama u JSON (engl. *JavaScript Object Notation)* formatu, ali korisnik također može koristiti i XML format. JSON je jednostavan tekstualni format za pohranu i prijenos podataka. Može se koristiti u bilo kojem programskom jeziku te se podaci mogu lako slati između računala.

```
let loc = document.getElementById("location");
let tempIcon = document.getElementById("temp-icon");
let tempValue = document.getElementById("temp-value");
let climate = document.getElementById("climate");
let iconFile;

window.addEventListener("load", () => {  //loads script
    let long;
    let lat;

    if (navigator.geolocation) {  // browser asks for location
    navigator.geolocation.getCurrentPosition((position) => {
        long = position.coords.longitude;
        lat = position.coords.latitude;
        const proxy ="https://cors-anywhere.herokuapp.com/"; // same-origin policy
        const api = `${proxy}api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${long}&appid=817230eb30c9fd78c7d884cfecd7dac2`;
        fetch(api) // gets api, make api call
            .then((response) => {
                return response.json();
            })
            .then(data => {
                console.log(data);
                const {name} = data; // object destructuring
                const {feels_like} = data.main;
                const {id, main} = data.weather[0];
                loc.textContent = name;
                climate.textContent = main;
                tempValue.textContent = Math.round(feels_like - 273) + 'ºC';
```

**Slika 5.9.** *Dio kôda za weather.js JavaScript datoteku.*

Za izdvajanja svojstava iz objekta koji su u JSON formatu te njihovo povezivanje s varijablama korišteno je destrukturiranje objekta. Preko te značajke se može pročitati svojstvo i dodijeliti njegova vrijednost bez dupliciranja naziva svojstva te se može izvući više svojstava iz istog objekta u samo jednoj naredbi.

Važno je napomenuti da se osim trenutne lokacije i vanjske temperature mijenja i slika vremenskog uvjeta ovisno o id-u koji je dohvaćen kada web preglednik ima pristup lokaciji uređaja.

Ako je id:

- Manji od 250, slika prikazuje oluju.
- Između 500 i 550, slika prikazuje kišu.
- Između 600 i 650, slika prikazuje snijeg.
- Između 700 i 750, slika prikazuje djelomično oblačno vrijeme.
- 800, slika prikazuje sunčano vrijeme.
- Veći od 800, slika prikazuje oblačno vrijeme.

## 5.4. Preporuke korisniku

U ovom radu osmišljeno je da su u prostorije kuće postavljena dva senzora, jedan u dnevnoj sobi te drugi u spavaćoj sobi. U realnom slučaju oni bi dohvaćali vrijednosti temperature i vlage unutar prostorija i automatski ih spremali u bazu podataka, ali za potrebe ovog rada podaci su upisivani ručno u *table_data* tablicu. Preporuke korisniku se daju na temelju posljednjeg podatka na senzorima te su napisane unutar *last_data.php* datoteke.

Idealna temperatura za dnevnu sobu je između 19 i 22 stupnja s postotkom vlage od 40 do 60 %, dok je za spavaću sobu idealna temperatura između 16 i 18 stupnjeva i relativnom vlažnosti zraka od 30 do 50 % [19]. Oba faktora utječu na zdravlje korisnika i namještaj unutar prostorija. Bitno je imati dobro izolirane vanjske zidove koji ne samo da sprječavaju gubitak topline, nego i povećavaju temperaturu na površini zida, a time i kvalitetu življenja. Preporučuje se svakodnevno provjetravanje prostorija u kraćim vremenskim intervalima te se treba držati vrata zatvorena između različito grijanih prostorija. Tuširanje, kuhanje i sušenje odjeće znatno povećava vlažnost u sobi. Zamagljivanje i kondenzacija koja se nakupila na prozorima, plijesan i vlaga na zidovima i stropovima pokazatelj su prevelike razine vlažnosti.

Višak vlage povećava razvoj plijesni i gljivica što uzrokuje zdravstvene probleme. Može prouzročiti truljenje i oštećenje stvari, ako se koristi ovlaživač zraka, treba ga isključiti. Za vrijeme kuhanja i kupanja preporučeno je otvoriti prozor ako vani ima svježeg i suhog zraka ili koristiti ispušne ventilatore. U suprotnom, niska vlaga uzrokuje statički elektricitet, povećanu osjetljivost na prehlade i respiratorne infekcije, te može omogućiti razvoj virusa i klica. Također oštećuje namještaj i elektroniku. Preporučuje se ostavljanje mokre odjeće i ručnika neko vrijeme da se suše u prostoriji, ali ne tijekom zime. Može se staviti i posuda za vodu pored radijatora ili kupiti prijenosni ovlaživač zraka radi povećanja količine vlage u sobi.

Navedene preporuke se daju unutar granica za razvoj plijesni, gljivica i kondenzacije koje su definirane na slici 5.10. [20]. X os predstavlja temperaturu u prostoru u Celzijevim stupnjevima, a Y os relativnu vlagu u prostoru izraženu u postotku.

**Slika 5.10.** *Granice za razvoj plijesni, gljivica i kondenzacije.*

Programski kôd na slici 5.11. prikazuje neke preporuke definirane za senzor jedan koji se nalazi u dnevnoj sobi. Za drugi senzor u spavaćoj sobi pojedine granice su drugačije zbog različito definirane idealne temperature i vlage. Postavljen je SQL upit koji dohvaća zadnju unesenu vrijednosti temperature i vlage na senzoru jedan iz tablice *table_data.* PHP *echo* naredba ispisuje poruku na zaslon.

```php
<?php
$query1 = "SELECT `temperature`, `humidity` FROM `table_data` WHERE `sensor` = 'sensor 1'
        ORDER BY `date_recorded` DESC LIMIT 1;";
$result = mysqli_query($connect, $query1);
$resultCheck = mysqli_num_rows($result);

if ($resultCheck > 0) {
    while($row = mysqli_fetch_assoc($result)) {
        if ($row['temperature'] > 22 && $row['temperature'] <= 35) {
            echo "[Sensor 1] Decrease your temperature to 19 - 22 °C." . '<br>';
        }
        if ($row['temperature'] > 35 && $row['temperature'] < 45) {
            echo "[Sensor 1] It is a bit warm. Decrease your temperature to at least 20 °C." . '<br>';
        }

        else if ($row['temperature'] < 19 && $row['humidity'] > 60) {
            echo "[Sensor 1] Increased chance of mold and mildew growth which can cause health problems.
                Excess moisture can cause rot, damaging your belongings.
                If you have a humidifier, turn it off.
                Use exhaust fans while cooking and bathing, or open a window if there is fresh, drier air outside." . '<br>';
        }
        else if ($row['temperature'] < 19 && $row['humidity'] <= 60) {
            echo "[Sensor 1] Increase your temperature to at least 19 °C." . '<br>';
        }
        else if ($row['temperature'] >= 19 && $row['temperature'] <= 22 && $row['humidity'] >= 40 && $row['humidity'] <= 60) {
            echo "<div class='green_text'>[Sensor 1] Temperature and humidity values in a room are ideal.</div>" . '<br>';
        }
        else if ($row['temperature'] >= 19 && $row['temperature'] <= 22 && $row['humidity'] > 60) {
            echo "[Sensor 1] Increased chance of mold and mildew growth which can cause health problems.
                Excess moisture can cause rot, damaging your belongings.
                If you have a humidifier, turn it off.
                Use exhaust fans while cooking and bathing, or open a window if there is fresh, drier air outside." . '<br>';
        }
```

**Slika 5.11.** *Pojedine preporuke za senzor jedan.*

## 5.5. Grafički prikaz vrijednosti

Registrirani korisnik može vidjeti grafički prikaz prosječnih vrijednosti temperature i vlage u određenom vremenskom intervalu. Uspoređuju se prosječne vrijednosti dobivene sa senzora unutar prostorija s vanjskim prosječnim vrijednostima te su prikazane na kombiniranom grafikonu (engl. *combo chart*) koji je spoj linijskog i stupčastog grafa. Korištena je Google Charts interaktivna web usluga za vizualizaciju podataka. Njoj se predaju podaci iz *table_data* i *outside_data* tablica na bazi podataka. Googleov grafički paket pruža *API* za umetanje svoje biblioteke u HTML.

Način na koji Google Charts funkcionira je sljedeći:

- *API* se poziva unutar <*head*> elementa u *header.php* datoteci.
- JavaScript izvršava Ajax zahtjev za *API.*

- *API* pretvara podatke iz tablica i definirane postavke u skalabilnu vektorsku grafiku (engl. *SVG*).
- *API* šalje natrag SVG.
- HTML prikazuje grafikon na web stranici.

Slika 5.12. prikazuje programski kôd za iscrtavanje kombiniranog grafa. *Google.charts.load()* funkcija učitava *API* i paket za vizualizaciju. *Google.charts.setOnLoadCallback()* pokreće funkciju *drawVisualization()* kada se učita *API*. Funkcija *drawVisualization1()* popunjava tablicu podacima iz baze podataka. Varijable *start_date* i *end_date* se odnose na početni i krajnji datum jer korisnik može filtrirati podatke u određenom vremenskom intervalu. SQL upit preko *GROUP BY* parametra grupira podatke po datumu.

```
<script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawVisualization1);
      google.charts.setOnLoadCallback(drawVisualization2);

      function drawVisualization1() {

        var data = google.visualization.arrayToDataTable([
          ['Date', 'Temperature inside', 'Temperature outside', 'Humidity inside', 'Humidity outside'],

          <?php

          require "registration/dbh.php";
          $start_date = $_POST['start_date_var'];
          $end_date = $_POST['end_date_var'];

          $query = "SELECT DATE(table_data.date_recorded) AS 'date_recorded', AVG(table_data.temperature) AS 'temp_inside',
          AVG(table_data.humidity) AS 'hum_inside', AVG(outside_data.temp_out) AS 'temp_out', AVG(outside_data.hum_out) AS 'hum_out'
          FROM table_data, outside_data
          WHERE (DATE(outside_data.date_recorded) = DATE(table_data.date_recorded)) AND (DATE(table_data.date_recorded) BETWEEN
          '$start_date'AND '$end_date') AND table_data.sensor = 'sensor 1'
          GROUP BY DATE(table_data.date_recorded);";

          $result = mysqli_query($connect, $query);
          $resultCheck = mysqli_num_rows($result);

          if ($resultCheck > 0) {
              while ($row = mysqli_fetch_assoc($result)) {
                      echo "['".$row['date_recorded']."', ".$row['temp_inside'].",
                              ".$row['temp_out'].", ".$row['hum_inside'].", ".$row['hum_out']."],";
              }
          }

        ?>
        ]);
```

**Slika 5.12.** *Programski kôd za iscrtavanje kombiniranog grafa.*

Na slici 5.13. se vide opcije grafikona. *Title* definira naslov grafa, *hAxis* je objekt koji predstavlja horizontalnu os gdje su raspoređeni dani, dok *vAxis* predstavlja vertikalnu os gdje su vrijednosti.

*Series* je niz objekata od kojih svaki opisuje njegov format na grafu. Nula i jedan su prvi i drugi objekt koji su zadani da budu linijski prikazani. Odnose se na temperaturu iznutra i izvana.

```
var options = {
  title : 'Average temperature and humidity values compared to average outside values on sensor 1',
  vAxis: {title: 'Values'},
  hAxis: {title: 'Days'},
  seriesType: 'bars',
  colors: ['FF1C0D', '#BB0000', '#5AA9FF', '#00E5FF'],
  series: {0: {type: 'line'}, 1: {type: 'line'}}
};

var chart = new google.visualization.ComboChart(document.getElementById('chart_div1'));
chart.draw(data, options);
```

**Slika 5.13.** *Opcije grafikona.*

## 5.6. Prikaz minimalne i maksimalne vrijednosti tijekom dana

Korisniku je omogućen i dnevni prikaz temperature i vlage unutar prostorija te njihove minimalne i maksimalne vrijednosti što može biti korisno ako neke vrijednosti izlaze izvan dopuštenih granica. Funkcionalnost je napisana unutar *one_date.php* datoteke. Koristi se *jQuery Datepicker* dodatak koji omogućava kalendarski prikaz datuma na web aplikaciji. On se koristi u svim dijelovima web aplikacije gdje korisnik ima mogućnost filtrirati podatke po datumu.

Slika 5.14. prikazuje programski kôd za dohvaćanje najmanje i najveće vrijednosti tijekom dana za senzor jedan. Istom logikom se dohvaćaju podaci i s drugog senzora. *MIN()* funkcija vraća najmanju vrijednost odabranog stupca, a *MAX()* funkcija vraća najveću. Naredba *AS* se koristi radi promjene imena stupca sa pseudonimom. Pseudonim postoji samo lokalno u SQL upitu. Funkcija *mysqli_query()* izvršava upit prema bazi podataka i prima dva parametra. Jedan je spoj na bazu podataka, a drugi SQL upit. *Mysqli_num_rows()* funkcija vraća broj redaka te se sprema u *resultCheck* varijablu koja provjerava ako ima ikakvih podataka u tablici. Funkcija *mysqli_fetch_assoc()* vraća asocijativni niz koji odgovara dohvaćenom retku te se sprema u *row* varijablu. Ako nema redaka, vraća *NULL* vrijednost. Gdjegod se dohvaćaju podaci iz baze koriste se navedene PHP funkcije osim kod prijave i registracije korisnika gdje se koriste pripremljene izjave objašnjene u poglavlju 5.2.

```php
<div class="row">
<div class="col-md-6">

<?php

$query1 = "SELECT MAX(temperature) AS `max_temperature`, MIN(temperature) AS `min_temperature`,
          MAX(humidity) AS `max_humidity`, MIN(humidity) AS `min_humidity`
          FROM `table_data` WHERE `sensor` = 'sensor 1' AND DATE(`date_recorded`) = '$one_date';";
$result = mysqli_query($connect, $query1);
$resultCheck = mysqli_num_rows($result);

if (isset($resultCheck) > 0) {
    while($row = mysqli_fetch_assoc($result)) {
        if ($row['max_temperature'] != '') {
        echo "Max temperature on sensor 1 is " .$row['max_temperature'] . '<br>' . '<br>';
        echo "Min temperature on sensor 1 is " .$row['min_temperature'] . '<br>' . '<br>';
        echo "Max humidity on sensor 1 is " .$row['max_humidity'] . '<br>' . '<br>';
        echo "Min humidity on sensor 1 is " .$row['min_humidity'] . '<br>' . '<br>';
        }
        else {
            echo "";
        }
    }
}

else {
    echo "No result";
}
```

**Slika 5.14.** *Dohvaćanje najmanje i najveće vrijednosti tijekom dana.*

## 5.7. Prikaz svih podataka sa senzora

Za prikaz i filtriranje svih unutarnjih podataka sa senzora korišten je objektno orijentiran način programiranja u PHP-u. U ostalim dijelovima kôda koristi se proceduralan način. Objektno orijentirano programiranje (*OOP*) se odnosi na stvaranje objekata koji sadrže podatke i funkcije, dok se proceduralno programiranje odnosi na pisanje procedura ili funkcija koje izvode operacije nad podacima. OOP ima prednost što se brže i lakše izvodi, pruža jasnu strukturu programa te je potrebno pisati manje kôda čime se olakšava održavanje, mijenjanje i otklanjanje pogrešaka, ali u nekim slučajevima zahtijeva više truda. Nije bitno koji se način programiranja koristi jer mogu dati isti rezultat.

HTML kôd za tablicu, polja za unos datuma i tipki je napisan unutar *datatable.php* datoteke. *Model.php* datoteka sadrži klasu Model koja ima varijable za povezivanje na bazu podataka te funkcije za dohvaćanje svih podataka iz *table_data* tablice i za odabir početnog i krajnjeg datuma. *Records.php* datoteka je povezana s *model.php* datotekom te provjerava da li su varijable za početni i krajnji datum postavljene.

Preko *json_encode()* funkcije kodira niz varijabli za datum u JSON format zato što je to, u ovom slučaju, potreban format za prikaz svih podataka u tablici.

Funkcionalnost za prikaz podataka sa senzora je na slici 5.15. *$.ajax()* metoda izvodi asinkroni Ajax zahtjev. Podaci u JSON formatu se šalju na bazu podataka POST metodom. Zahtjev se šalje na *records.php* datoteku kao što je navedeno unutar *url-a.*

```
function data_fetch(start_date, end_date) {
    $.ajax({
      url: "records.php",
      type: "POST",
      data: {
          start_date: start_date,
          end_date: end_date
      },
      dataType: "json",
      success: function(data) {
          // Datatables

          $('#records').DataTable( {
            "data": data,
            "responsive": true,
            "columns": [{
                          "data": "date_recorded",
                        },
                        {
                          "data": "temperature",
                        },
                        {
                          "data": "humidity",
                        },
                        {
                          "data": "sensor"
                        }
            ]
        });
      }
    });
  }

  data_fetch();
```

**Slika 5.15.** *jQuery funkcija za prikaz podataka sa senzora.*

Korisnik može filtrirati podatke. Potrebno je odabrati oba datuma što se vidi iz slike 5.16. *Destroy()* metoda uklanja postojeću tablicu koja po zadanim postavkama prikazuje sve podatke sa senzora te ju zamjenjuje drugom tablicom koja prikazuje podatke u vremenskom intervalu koje je korisnik odabrao. Uklanjanje tablice je potrebno kako bi se spriječilo propuštanje ili „curenje“ memorije.

```
// Filter button
$(document).on("click", "#filter1_btn", function(e) {
  e.preventDefault();

  var start_date = $("#start_date1").val();
  var end_date = $("#end_date1").val();

  if (start_date == "" || end_date == "") {
    alert("Both date required");
  }
  else {
    $('#records').DataTable().destroy();
    data_fetch(start_date, end_date);
  }
});
```

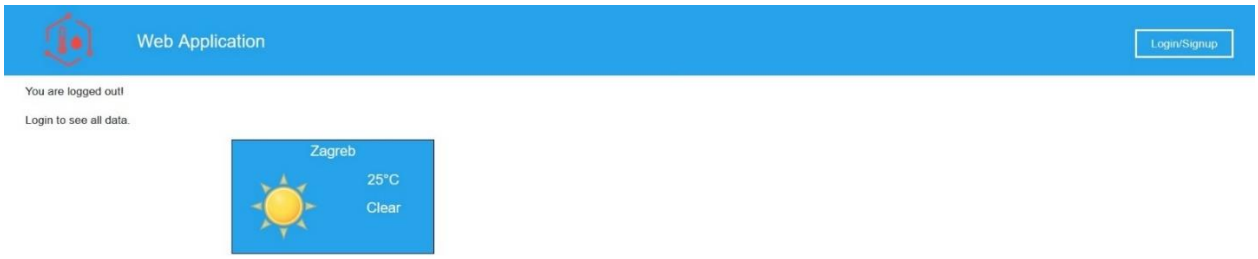
**Slika 5.16.** *Funkcija za odabir oba datuma.*

# 6. KORIŠTENJE APLIKACIJE

Za pokretanje web aplikacije potrebno je preuzeti i instalirati XAMPP serverski paket. XAMPP simulira rad servera te preko *xampp-control* aplikacije je potrebno uključiti Apache web server i MySQL modul na kojem se nalazi *webapp* baza podataka. Web aplikaciju je potrebno spremiti u *htdocs* datoteku te se pokreće u web pregledniku. U adresnu traku se upisuje *localhost* te se odabire naziv projekta ili nakon upisa se napiše puno ime datoteke (npr. *localhost/web_application*).

## 6.1. Neregistrirani korisnik

Korisnik koji nije prijavljen na stranicu može vidjeti vrijednost temperature izvana, njen kratak opis i sličicu koja prikazuje kakvo je vrijeme. Ti podaci se mijenjaju svaki put kada korisnik web pregledniku da pristup lokaciji uređaja. Iznad tih podataka vidi lokaciju mjesta u kojem se nalazi. Na stranici može vidjeti i poruku koja kaže da se treba prijaviti ako želi vidjeti ostale podatke. Za prijavu je potrebno kliknuti na gumb „*Login/Signup*“ koji se nalazi na navigacijskoj traci. Slika 6.1. prikazuje izgled web aplikacije za neprijavljenog korisnika.



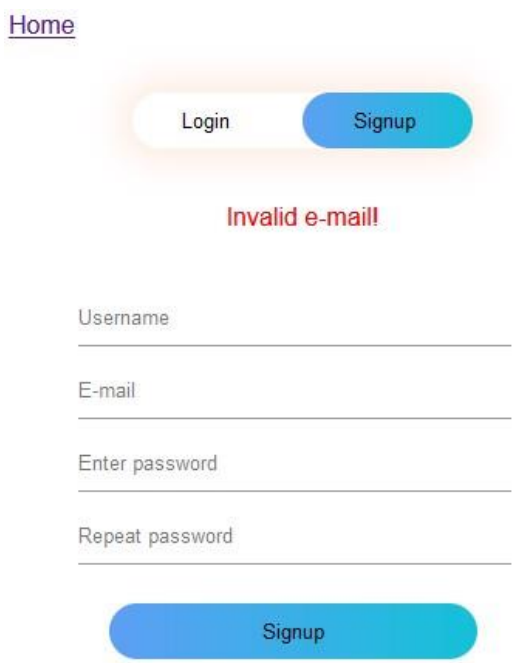

**Slika 6.1.** *Izgled web aplikacije za neprijavljenog korisnika.*

Nakon klika na „*Login/Signup*“ otvara se forma za prijavu i registraciju korisnika. Provjere za registraciju novog korisnika su sljedeće:

- Potrebno je ispuniti sva polja za unos.
- E-mail adresa mora biti važeća.
- Lozinka i ponovljena lozinka moraju biti jednake.

- Korisničko ime ne smije imati specijalne znakove, mora se sastojati od slova i brojeva ili kombinacije slova i brojeva.
- Korisničko ime mora biti jedinstveno u bazi podataka.

Slika 6.2. pokazuje slučaj kada je unesena neispravna e-mail adresa.




**Slika 6.2.** *Neispravna e-mail adresa u formi za registraciju.*

Nakon što se popune svi podaci, klikom na „*Signup*“ tipku korisnički račun će biti uspješno napravljen. Nakon toga je potrebno kliknuti na „*Login*“ čime se otvara forma za prijavu korisnika. Forma sadrži polje za unos korisničkog imena i lozinke te provjerava je li su sva polja popunjena i ispravnost korisničkog imena i lozinke.

## 6.2. Registrirani korisnik

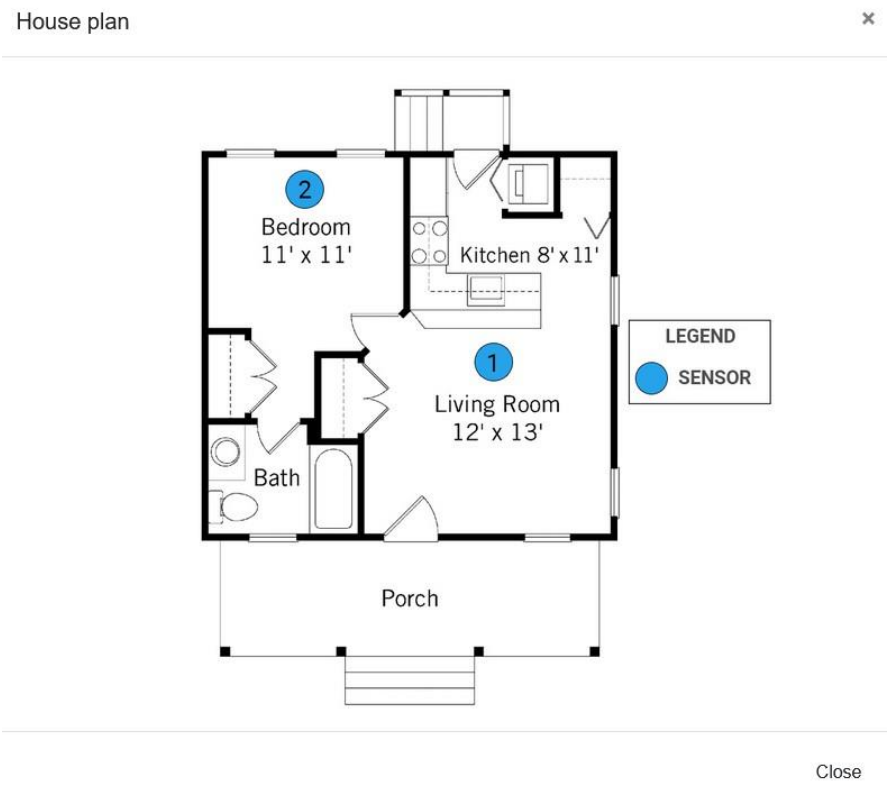
Registrirani korisnik ima mogućnost vidjeti sve podatke na web aplikaciji. Slika 6.3. prikazuje sve registrirane korisnike u bazi podataka u *users* tablici gdje se vidi da je lozinka raspršena zbog sigurnosti, dok slika 6.4. prikazuje izgled početne stranice nakon uspješne prijave.

| | idUsers | uidUsers | emailUsers | pwdUsers |
|---|---|---|---|---|
| Edit Copy Delete | 1 | Student | student@gmail.com | $2y$10$hcrzAjMNsGJse1DkucW.0uuQDDvJ0Cn3T18F8YDPkqn... |
| Edit Copy Delete | 2 | Bernardo | bernardo@gmail.com | $2y$10$zFr7/oACQQMZDBV60z7d5OmidykA5KUE.it8ZYvRvfT... |

**Slika 6.3.** *Prikaz svih registriranih korisnika u bazi podataka.*

**Slika 6.4.** *Izgled početne stranice nakon uspješne prijave.*

Klikom na „*Logout*“ gumb koji se nalazi na navigacijskoj traci, korisnik se uspješno odjavljuje iz web aplikacije. Ako želi vidjeti tlocrt kuće s pozicijom senzora kliknuti će na gumb „*House plan*“. Slika 6.5. prikazuje tlocrt kuće sa senzorom jedan koji se nalazi u dnevnoj sobi, a senzor dva u spavaćoj sobi. Skočni prozor se zatvara klikom na gumb „*X*“ ili „*Close*“.

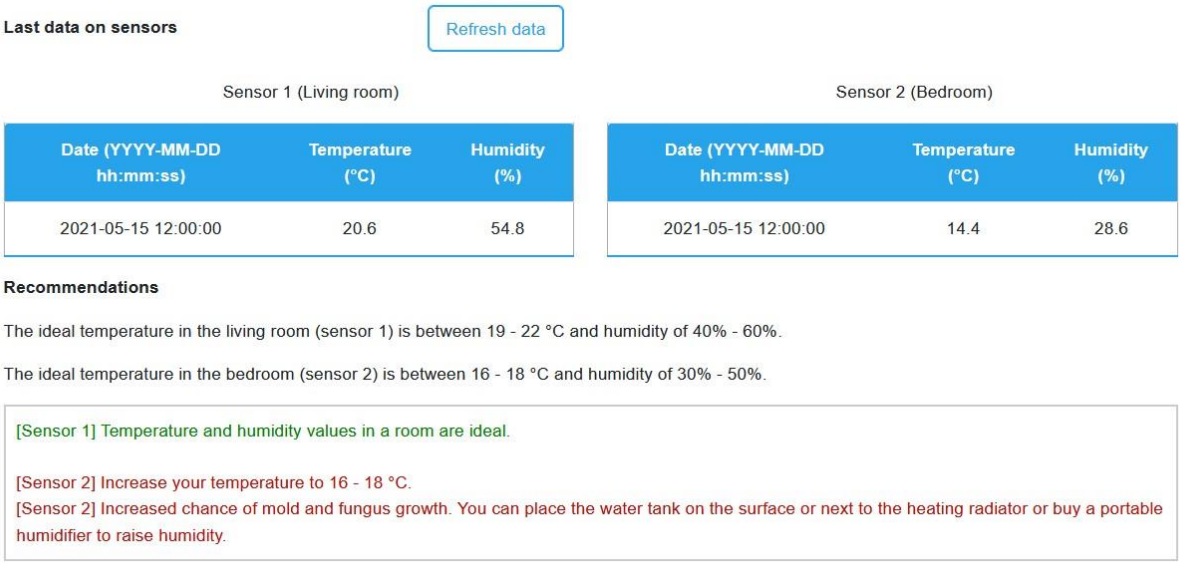



**Slika 6.5.** *Tlocrt kuće s pozicijom senzora.*

Klikom na „*All data*“ otvara se nova stranica na kojoj je tablica koja sadrži sve podatke sa senzora koji mjere temperaturu i razinu vlažnosti unutar kuće. Korisnik ima brojne mogućnosti poput sortiranja podataka između dva datuma, resetiranja tablice u početno stanje, traženja određenog podatka te mijenjane redoslijeda svih podataka od najvećeg prema najmanjem ili obrnuto. Početno stanje tablice prikazuje podatke od prvog unesenog podatka prema zadnjem. Slika 6.6. prikazuje prikaz svih podataka koji su uneseni u četvrtom mjesecu 2021. godine. Korisnik može vidjeti 10, 25, 50 ili 100 podataka odjednom ili može koristiti tipke ispod tablice za prikaz podataka. Strelice između svakog stupca omogućuju mijenjane redoslijeda podataka. Datum završetka ne može biti prije datuma početka te oba datuma moraju biti unesena za filtriranje podataka.

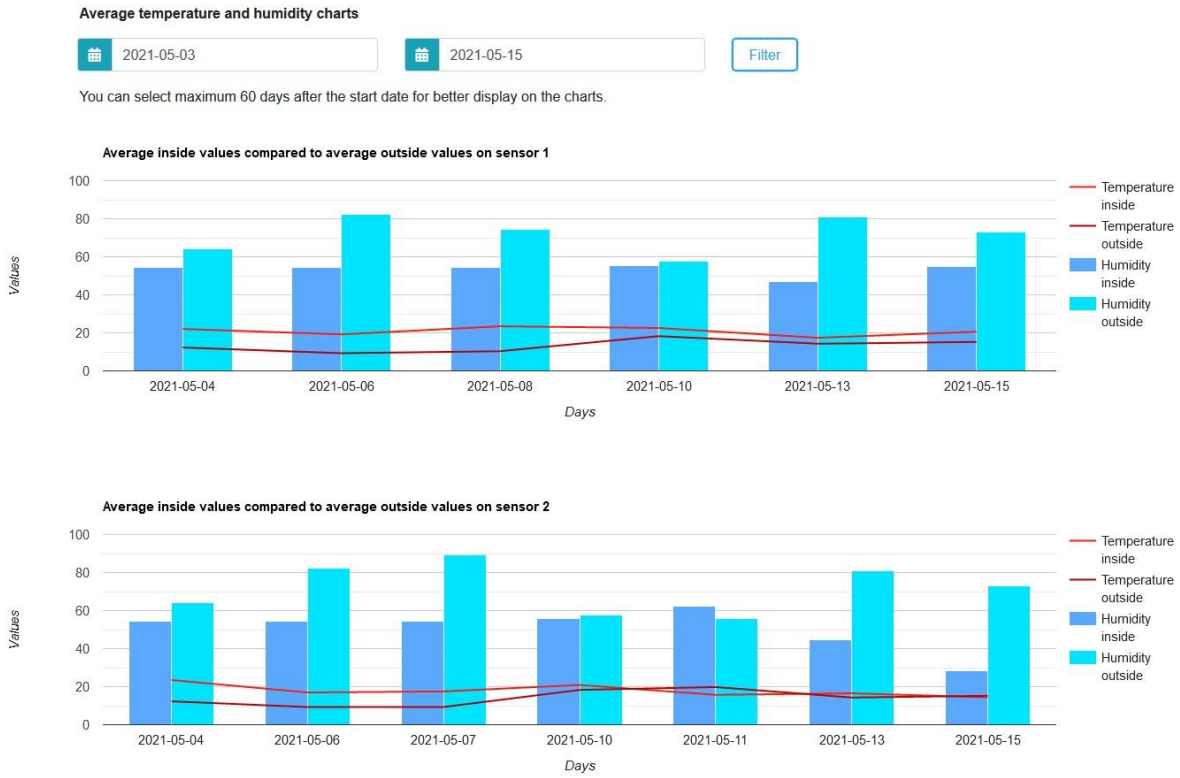
**Slika 6.6.** *Prikaz svih unutarnjih podataka u četvrtom mjesecu 2021. godine.*

Klikom na „*Home*“ korisnik se vraća na početnu stranicu. Sljedeće što vidi je prikaz posljednjeg podatka s oba senzora te preporuke što treba napraviti ako podaci nisu unutar idealnih granica. Slika 6.7. prikazuje slučaj kada su posljednji podaci na senzoru u dnevnoj sobi unutar idealnih granica, dok na senzoru u spavaćoj sobi nisu. Klikom na „*Refresh data*“ se ponovno učitavaju samo zadnji podaci te tako nije potrebno učitati ili „osvježiti“ cijelu stranicu.



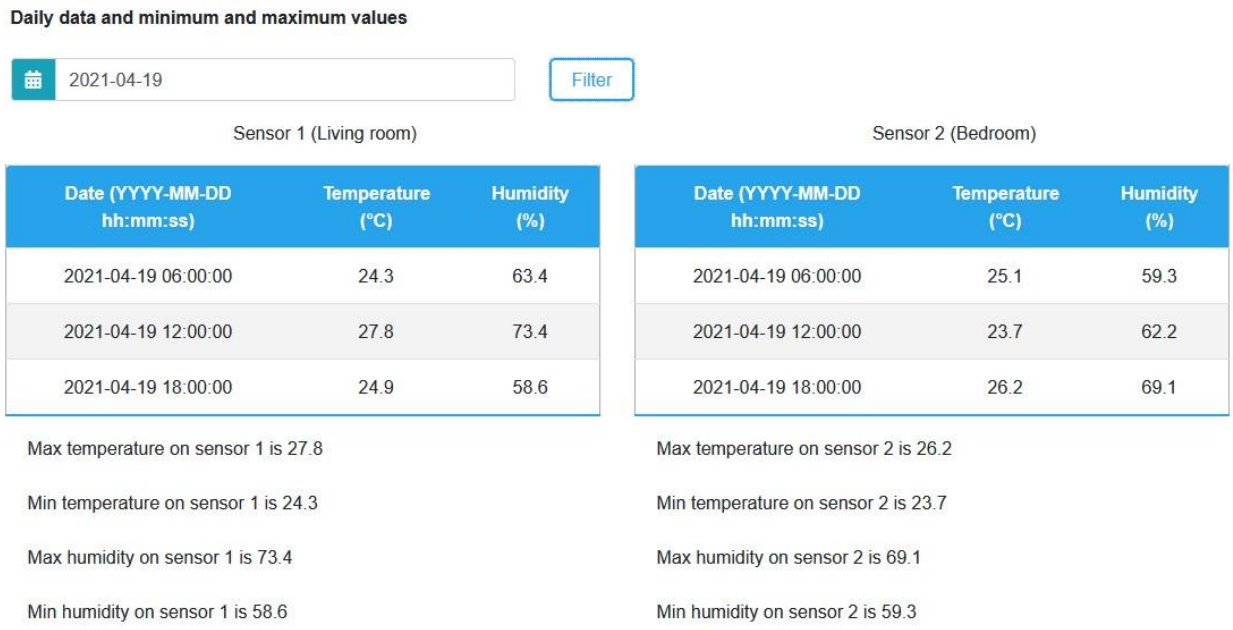

**Slika 6.7.** *Prikaz posljednjeg podatka na senzorima i preporuke korisniku.*

Ispod preporuka, korisnik ima mogućnost vidjeti usporedbu prosječnih vrijednosti temperature i vlage unutar i izvan kuće. Vrijednosti su prikazane na kombiniranome grafikonu. To je jedino mjesto na web aplikaciji gdje se koriste vanjske vrijednosti temperature i vlage koje se dohvaćaju iz *outside_data* tablice. Kao i kod prikaza svih unutarnjih podataka sa senzora, korisnik treba odabrati početni i krajnji datum za filtriranje podataka. Datum završetka ne može biti prije datuma početka te je moguće najviše odabrati 60 dana nakon početnog datuma radi bolje preglednosti na grafikonima. Slika 6.8 prikazuje uspoređene prosječne vrijednosti. Crvena linija predstavlja vrijednosti temperature izvana, dok narančasta linija predstavlja vrijednosti temperature iznutra. Stupac obojan tirkiznom bojom predstavlja vanjske vrijednosti vlage, a tamno plavom bojom su obojane unutarnje vrijednosti vlage. Korisnik može postaviti pokazivač miša na liniju ili stupac da vidi prosječnu vrijednosti temperature ili vlage tijekom određenog dana.




**Slika 6.8.** *Grafički prikaz prosječnih unutarnjih i vanjskih vrijednosti.*

Posljednje što korisnik može vidjeti je dnevni prikaz unutarnjih podataka sa senzora te njihove najmanje i najveće vrijednosti. Potrebno je odabrati jedan datum za prikaz vrijednosti. U slučaju da nema podataka za određeni dan korisniku će se ispisati poruka „Nema rezultata na senzoru“. Dnevni prikaz vrijednosti je na slici 6.9.




**Slika 6.9.** *Dnevni prikaz unutarnjih vrijednosti sa senzora.*

# 7. ZAKLJUČAK

U sklopu ovog završnog rada, napravljena je web aplikacija za prikaz i praćenje vrijednosti temperature i vlage unutar kuće. Korisnik se treba prijaviti kako bi mogao vidjeti sve podatke. Uključen je sigurnosni mehanizam kojim se smanjuje mogućnost zloupotrebe računa jer se u bazi podataka lozinka šifrirana. Napravljen je jednostavni i pregledni prikaz osnovnih podataka sa senzora unutar kuće. Pri potrebi detaljnih informacija, korisnik ima mogućnost grafičkog prikaza usporedbe prosječnih unutarnjih i vanjskih vrijednosti. Na stranici su određene smjernice koje olakšavaju korištenje web aplikacije korisnicima koji imaju skromnije informatičko znanje.

Problem vlage u kućanstvu povećava financijske troškove, štetu namještaja te ljudsko zdravlje dovodi u pitanje. Prednost aplikacije napravljene u ovom radu omogućava korisniku uvid i preventivno djelovanje tako da do problema ne bi došlo.

Prvi korak unapređenju web aplikacije bio bi slanje obavijesti na nekom većem odstupanju od idealnih granica. Sljedeći korak bio bi da se u sustav dodaju izvršni elementi koji bi automatskim procesom, a ne samo preporukom, odradili potrebne radnje kako bi se spriječili problemi.

# LITERATURA


[1] Cras Telecontrol Service, http://cras.hr/company/cts.html 22.5.2021.

[2] Humidex App, https://humidex.com/en/humidex-myhome-en 22.5.2021.

[3] EDGEVue App, https://temptimecorp.com/temperature-indicators-sensors/edge-electronic-sensors/edgevue-app/ 22.5.2021.

[4] Definition of Web Application, https://blog.stackpath.com/web-application/ 24.5.2021.

[5] Web Application Based Websites, https://www.indiamart.com/proddetail/web-application-based-websites-4416777730.html 24.5.2021.

[6] Frontend vs Backend Development, https://careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend/ 26.5.2021.

[7] Difference between Frontend and Backend, https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end 26.5.2021.

[8] HTML Basics, https://www.hostinger.com/tutorials/what-is-html 14.6.2021.

[9] Differences Between HTML and HTML5, https://www.hostinger.com/tutorials/difference-between-html-and-html5 14.6.2021.

[10] Difference Between CSS vs CSS3, https://www.educba.com/css-vs-css3/ 25.6.2021.

[11] Bootstrap: Getting Started, https://getbootstrap.com/docs/5.0/getting-started/introduction/ 28.6.2021.

[12] JavaScript Definition, https://skillcrush.com/blog/javascript/ 3.7.2021.

[13] Ajax Programming, https://en.wikipedia.org/wiki/Ajax_(programming) 16.7.2021.

[14] PHP: Getting Started, https://www.php.net/manual/en/intro-whatcando.php 5.7.2021.

[15] Difference between SQL and NoSQL, https://www.geeksforgeeks.org/difference-between-sql-and-nosql/ 16.7.2021.

[16] MySQL Tutorial, https://www.digitalocean.com/community/tutorials/what-is-mysql 29.6.2021.

[17] Developer Survey Results 2019, https://insights.stackoverflow.com/survey/2019#development-environments-and-tools 30.6.2021.

[18] Weather Underground, https://www.wunderground.com/ 1.7.2021.
[19] Optimalna sobna temperatura, https://nobel.ba/blog/svijet-grijanja/optimalna-sobna-temperatura-atlantic-grijanje-120/ 15.8.2021.
[20] Savjeti stručnjaka, https://www.jutarnji.hr/domidizajn/opasna-plijesan-na-zidovima-8270102 15.8.2021.

# SAŽETAK

Cilj ovog rada izraditi je web aplikaciju za prikaz vrijednosti temperature i vlage pojedinih prostorija unutar kuće. Na početku su opisane frontend i backend tehnologije koje su bile potrebne za izradu ove aplikacije. Najviše je korišten PHP u kombinaciji s Ajax tehnologijom. Svi podaci se spremaju na MySQL bazu podataka. Aplikacija se pokreće preko web preglednika koristeći XAMPP serverski paket. Potrebno se prijaviti radi pregledavanja vrijednosti. Ovisno o vrijednostima očitanih sa senzora, korisniku se daju preporuke što treba učiniti da vrijednosti održi unutar preporučenih granica. Korisnik ima uvid u dnevno očitanje s najmanjom i najvećom vrijednosti tijekom dana te može sortirati vrijednosti unutar određenog razdoblja. Omogućen je prikaz vremenske prognoze te usporedba prosječnih vrijednosti unutar i izvan kuće.

**Ključne riječi:** baza podataka, internet stvari, temperatura, vlaga, web aplikacija

# ABSTRACT

**Title:** Web application for collecting and displaying data from an IoT station

The goal of this thesis is to develop a web application that displays temperature and humidity values inside specific rooms in the house. At the beginning of this work, frontend and backend technologies that were used to create this application have been shortly described. PHP is usually used in combination with Ajax technology. All data is stored in a MySQL database. Application starts through a web browser using the XAMPP cross-platform web server solution stack package. It is necessary to log in to check the values. Depending on the read values from the sensor, the user is given recommendations to keep those values inside the recommended range. User can see daily data with the lowest and highest values during the day and can sort the values within a certain period. It is enabled to display the weather forecast and compare the average values inside and outside of a house.

**Key words:** data base, humidity, internet of things, temperature, web application

# ŽIVOTOPIS

Bernardo Kopjar rođen je 17. travnja 1995. godine u Osijeku. Odrastao je u Valpovu, gdje i danas živi. Nakon osnovne škole, u istom gradu upisuje opću gimnaziju. Svoje obrazovanje nastavlja upisom stručnog studija Informatike na Fakultetu Elektrotehnike, Računarstva i Informacijskih tehnologija u Osijeku. Na 3. godini studija odrađuje praksu u Atos tvrtki gdje unapređuje znanja o frontend i backend tehnologijama. 2021. godine pohađa akademiju za kibernetičku sigurnost radi daljnjeg educiranja.

# PRILOZI

Projektna mapa s izvornim kôdom i priloženim word dokumentom nalazi se na optičkom disku koji je priložen uz ispisanu verziju završnog rada.