



# SADRŽAJ

|  |          |
|--|----------|
| <b>1. Uvod</b>                                       | <b>1</b> |
| <b>2. Arhitektura i dizajn sustava</b>               | <b>2</b> |
| 2.1. Opća arhitektura sustava . . . . .              | 2        |
| 2.2. Dizajn korisničkog sučelja . . . . .            | 2        |
| 2.3. Komunikacija s WeatherAPI-jem . . . . .         | 2        |
| 2.4. Upravljanje bazom podataka . . . . .            | 3        |
| 2.5. Sigurnost i zaštita podataka . . . . .          | 3        |
| <b>3. Implementacija</b>                             | <b>4</b> |
| 3.1. Izgradnja klijentske aplikacije . . . . .       | 4        |
| 3.1.1. Struktura koda . . . . .                      | 4        |
| 3.1.2. Glavna komponenta . . . . .                   | 4        |
| 3.1.3. Automatizirana obrada podataka . . . . .      | 5        |
| 3.2. Izgradnja poslužiteljske aplikacije . . . . .   | 5        |
| 3.2.1. Korištene tehnologije i biblioteke . . . . .  | 5        |
| 3.2.2. Struktura koda i zadaća direktorija . . . . . | 5        |
| 3.2.3. Povezivanje s WeatherAPI-jem . . . . .        | 6        |
| 3.2.4. Preusmjeravanje . . . . .                     | 6        |
| 3.2.5. Tokeni . . . . .                              | 6        |
| 3.2.6. Povezivanje s bazom podataka . . . . .        | 7        |
| 3.3. Interakcija s WeatherAPI sučeljem . . . . .     | 7        |
| 3.4. Baza podataka . . . . .                         | 7        |
| 3.4.1. Tablica Users . . . . .                       | 8        |
| 3.4.2. Tablica Locations . . . . .                   | 9        |
| 3.4.3. Tablica Refresh Tokens . . . . .              | 9        |
| 3.5. Testiranje i ispravljanje pogrešaka . . . . .   | 10       |
| 3.5.1. Postman . . . . .                             | 10       |

|  |           |
|--|-----------|
| 3.5.2. VS Code i evidentiranje grešaka . . . . . | 10        |
| <b>4. Zaključak</b>                              | <b>12</b> |
| <b>Literatura</b>                                | <b>13</b> |

# 1. Uvod

Vremenska prognoza igra ključnu ulogu u našim svakodnevnim životima, utječući na odluke koje se kreću od odabira odjeće do planiranja putovanja. Preciznost i dostupnost vremenskih prognoza postale su sve važnije s rastućim utjecajem klimatskih promjena. Stoga, postoji potreba za pouzdanim i pristupačnim sustavima za praćenje vremenskih prognoza. Završni rad fokusira se na oblikovanje i izgradnju takvog sustava. Cilj je razviti korisničko sučelje koje će na čitljiv i interaktivan način prikazivati podatke o vremenskoj prognozi. Podatke će dohvaćati putem aplikacijskog programskog sučelja WeatherAPI [7] (*engl. API - Application Programming Interface*) što omogućava pružanje ažuriranih informacija u stvarnom vremenu. Osim prikaza trenutne vremenske prognoze za korisnikovu lokaciju, sustav će omogućiti pretraživanje drugih lokacija te pružiti predviđanje vremenske prognoze za odabrane lokacije. Ova funkcionalnost omogućava korisnicima da planiraju događaje u skladu s očekivanim vremenskim uvjetima. Kroz ovaj rad demonstrirat će se proces oblikovanja i izgradnje ovog sustava, kao i razmatranje mogućih izazova i rješenja pronađenih tijekom njegove izrade.

## **2. Arhitektura i dizajn sustava**

Arhitektura aplikacije sastoji se od web poslužitelja, web aplikacije i baze podataka te predstavlja ključni dio proizvoda. Ovaj tip arhitekture omogućuje razdvajanje odgovornosti na tri glavna dijela sustava što pomaže u poboljšanju performansi, skalabilnosti i sigurnosti aplikacije. Osim navedenog, aplikacija koristi aplikacijsko programsko sučelje WeatherAPI pomoću kojeg dohvaća redovno ažurirane podatke o trenutnom vremenu i vremenskoj prognozi.

### **2.1. Opća arhitektura sustava**

Sustav je projektiran kao klijent-poslužitelj aplikacija. Klijentska strana, koja se pokreće u korisnikovom web pregledniku, razvijena je koristeći React Typescript [5]. Klijent komunicira s poslužiteljem putem HTTP zahtjeva za dohvat vremenskih podataka. Poslužiteljska strana, razvijena u Pythonu, služi kao posrednik između klijenta i aplikacijskog programskog sučelja WeatherAPI. Kada poslužitelj primi zahtjev od klijenta, šalje zahtjev sučelju, obrađuje odgovor i šalje podatke natrag klijentu.

### **2.2. Dizajn korisničkog sučelja**

Korisničko sučelje osmišljeno je s ciljem jednostavnosti i čitljivosti. Glavni zaslon prikazuje trenutnu vremensku prognozu za korisnikovu lokaciju. Korisnik može unijeti ime drugog grada u pretraživačku traku za prikaz vremenske prognoze za tu lokaciju.

### **2.3. Komunikacija s WeatherAPI-jem**

Komunikacija sa aplikacijskim programskim sučeljem WeatherAPI ostvarena je kroz HTTP GET/POST zahtjeve. Zahtjevi su poslani s poslužiteljske strane koristeći Pythonov requests modul. Odgovori od API-ja obrađeni su i proslijeđeni klijentskoj strani

u obliku JSON odgovora.

## **2.4. Upravljanje bazom podataka**

Baza podataka, implementirana koristeći PostgreSQL, koristi se za spremanje podataka o korisnicima. Kada korisnik prvi put pristupi sustavu generira se token pomoću kojega korisnik kasnije pristupa uslugama aplikacije.

## **2.5. Sigurnost i zaštita podataka**

S obzirom na prirodu podataka koje sustav obrađuje, implementirane su osnovne mjere zaštite podataka. Svi podaci preneseni između klijenta, poslužitelja i API-ja šifrirani su koristeći HTTPS protokol. Dodatno, baza podataka zaštićena je od SQL injekcija korištenjem parametriziranih upita.

## 3. Implementacija

### 3.1. Izgradnja klijentske aplikacije

Klijentska aplikacija izgrađena je koristeći React Typescript, s fokusom na modularnost i održivost. Cjelokupna aplikacija organizirana je oko centralne App.tsx komponente koja upravlja rutama i navigacijom između različitih stranica.

#### 3.1.1. Struktura koda

Glavna struktura koda organizirana je na sljedeći način:

- **App.tsx**: Ovo je glavna komponenta aplikacije. Uključuje preusmjeravanje do različitih stranica i kontrolu pristupa ovisno o tome je li korisnik prijavljen ili nije.
- **models**: Ovaj direktorij sadrži modele koji se koriste za automatsko spremanje podataka s poslužitelja.
- **pages**: Sadrži sve stranice koje su dostupne u aplikaciji. Svaka stranica je smještena u vlastiti direktorij.
- **components**: Ovaj direktorij sadrži komponente koje su dio svake stranice, kao što su zaglavlje (*engl. header*) i podnožje (*engl. footer*).

#### 3.1.2. Glavna komponenta

Glavna App.tsx komponenta upravlja rutama i navigacijom kroz cijelu aplikaciju koristeći react-router-dom [4] paket. Na temelju dostupnosti pristupnog tokena (*engl. access token*) u sesiji, određuje se prikazuje li se standardno zaglavlje ili zaglavlje koje je prilagođeno za prijavljene korisnike. Svaka ruta asocirana je s određenom komponentom koja se prikazuje kada je ta ruta aktivna.

### 3.1.3. Automatizirana obrada podataka

Direktorij "models" ima ključnu ulogu u obradi podataka koje klijentska aplikacija prima od poslužitelja. Svaka datoteka unutar ovog direktorija predstavlja model koji odgovara strukturi podataka koju poslužitelju vraća sučelje WeatherAPI za određene vremenske uvjete. Modeli su oblikovani tako da točno odgovaraju strukturi JSON odgovora koji se dobiva od poslužitelja. To omogućava aplikaciji da automatski mapira podatke u modele bez potrebe za ručnom obradom. Tako dobivenim informacijama možemo lako i učinkovito manipulirati unutar aplikacije. Korištenjem ovih modela, aplikacija može automatski obraditi vrlo složene i detaljne podatke koje vraća poslužitelj, a istovremeno se održava čitljivost i modularnost koda. Ovo drastično smanjuje količinu potrebnog koda za obradu podataka što čini cijelu aplikaciju efikasnijom i lakšom za održavanje.

## 3.2. Izgradnja poslužiteljske aplikacije

### 3.2.1. Korištene tehnologije i biblioteke

Cijeli poslužiteljski dio naše web aplikacije izgrađen je koristeći Python-snažan, fleksibilan i lako čitljiv programski jezik. Izabrali smo Flask [1], mikro web okvir napisan u Pythonu, za strukturu naše poslužiteljske aplikacije. Flask nam omogućava jednostavnu implementaciju i prilagodbu različitih aspekata naše aplikacije.

Za rad s bazom podataka koristili smo Flask-SQLAlchemy, dodatak za Flask koji pruža SQLAlchemy podršku. SQLAlchemy nam omogućava da modeliramo tablice baze podataka kao Python klase.

Koristili smo Flask-JWT-Extended za autentifikaciju korisnika. Ovaj dodatak omogućava upotrebu JSON Web Tokena (JWT) za autentifikaciju i pruža sigurno upravljanje tokenima.

Za formiranje i validaciju unesenih podataka koristili smo Flask-WTF, dodatak koji olakšava rad s podacima koje je unio korisnik unutar Flask aplikacije.

### 3.2.2. Struktura koda i zadaća direktorija

Glavne komponente naše poslužiteljske aplikacije su:

- **config.py**: Datoteka koja sadrži konfiguracijske parametre za našu aplikaciju. Konfiguracija uključuje postavke za različita okruženja (razvojno, produkcijsko, testno), putanje do baze podataka, tajne ključeve za JWT, itd.



- **database:** Direktorij namijenjen za upravljanje bazom podataka. Sadrži skripte za inicijalizaciju baze podataka i definiranje sheme. Koristimo je i pri komunikaciji s bazom podataka.
- **forms:** Direktorij sadrži strukturu svih zahtjeva koji mogu biti poslani s klijent-ske strane kao što su obrazac za prijavu ili obrazac za registraciju.
- **models:** Direktorij sadrži definicije modela podataka. Modeli su preslike tablica u bazi podataka koje olakšavaju rad s podacima unutar aplikacije.
- **routes:** Direktorij sadrži definicije svih krajnjih točki (*engl. endpoints*) u našem poslužitelju. Svaka ruta je definirana u skladu s REST arhitekturom te uključuje definiciju metode (GET, POST, itd.), logiku obrade te odgovarajući odgovor.
- **\_\_init\_\_.py:** Skripta koja inicijalizira našu Flask aplikaciju, a uključuje potrebne module i postavlja konfiguraciju. Osim toga, ovdje se također definira i pokreće naša aplikacija.

Ovakvim strukturiranjem aplikacija ostaje čitljiva i lako održiva, a modularnost omogućava efikasno skaliranje i razvoj.

### 3.2.3. Povezivanje s WeatherAPI-jem

Podatke o vremenskim uvjetima dobivamo putem aplikacijskog programskog sučelja WeatherAPI. U našem kodu koristimo Python requests biblioteku za slanje HTTP zahtjeva prema WeatherAPI-ju. Odgovor WeatherAPI-ja zatim obrađujemo i koristimo za pružanje informacija o vremenu našim korisnicima.

### 3.2.4. Preusmjeravanje

Koristimo Flaskov ugrađen sustav za preusmjeravanje kako bismo definirali način na koji naša aplikacija reagira tijekom klijentskih zahtjeva na određenoj putanji web adrese. Svaka ruta je povezana s Python funkcijom koja se izvodi kada klijent posjeti tu putanju.

### 3.2.5. Tokeni

Za autentifikaciju korisnika koristimo JSON Web Tokene (JWT). Tijekom postupka prijave generiraju se dva tokena - pristupni token (*engl. access token*) i token za obnavljanje (*engl. refresh token*).

Pristupni token se koristi za autentifikaciju korisnika tijekom svakog zahtjeva na server.

On ima kraće vrijeme trajanja i koristi se za provjeru identiteta korisnika. Obnavlja jući token ima duže vrijeme trajanja i pohranjuje se u bazi podataka. Njegova je svrha omogućiti generiranje novog pristupnog tokena kada trenutni istekne bez potrebe za ponovnom prijavom korisnika. Ova dva tokena zajedno omogućuju siguran i učinkovit sustav autentifikacije korisnika. Za upravljanje ovim JWT tokenima koristimo Flask-JWT-Extended biblioteku koja omogućuje generiranje, primanje i provjeru JWT tokena.

### 3.2.6. Povezivanje s bazom podataka

Naša aplikacija koristi PostgreSQL [2] bazu podataka za pohranu podataka. Povezivanje s bazom podataka postiže se kroz SQLAlchemy objektno-relacijsko mapiranje (*engl. ORM - Object-Relational Mapping*) za Python. SQLAlchemy nam omogućava da interakciju s bazom podataka obavljamo na visokoj razini apstrakcije jer radimo s Python objektima umjesto izravnim SQL upitima. Definirali smo modele, koristeći SQLAlchemy, koji predstavljaju tablice unutar naše baze podataka i koristimo se njima za stvaranje, dohvaćanje, ažuriranje i brisanje podataka.

## 3.3. Interakcija s WeatherAPI sučeljem

Sučelje WeatherAPI<sup>1</sup> pruža trenutne vremenske podatke kao i vremensku prognozu za narednih 14 dana, povijesne podatke o vremenu i geografske podatke putem REST API-ja u JSON formatu. Sučelje također pruža informacije o vremenskim zonama, astronomskim podacima i geografskim podacima. Podatci o vremenu prikupljaju se u suradnji s nekoliko vladinih i meteoroloških agencija.

Za dohvat vremenskih podataka koristi se aplikacijsko programsko sučelje WeatherAPI. Komunikacija sa sučeljem odvija se pomoću HTTP GET zahtjeva koje poslužiteljska skripta postavlja koristeći Pythonov requests modul. Povratni podaci se zatim pretvaraju u JSON format prije slanja klijentskoj aplikaciji.

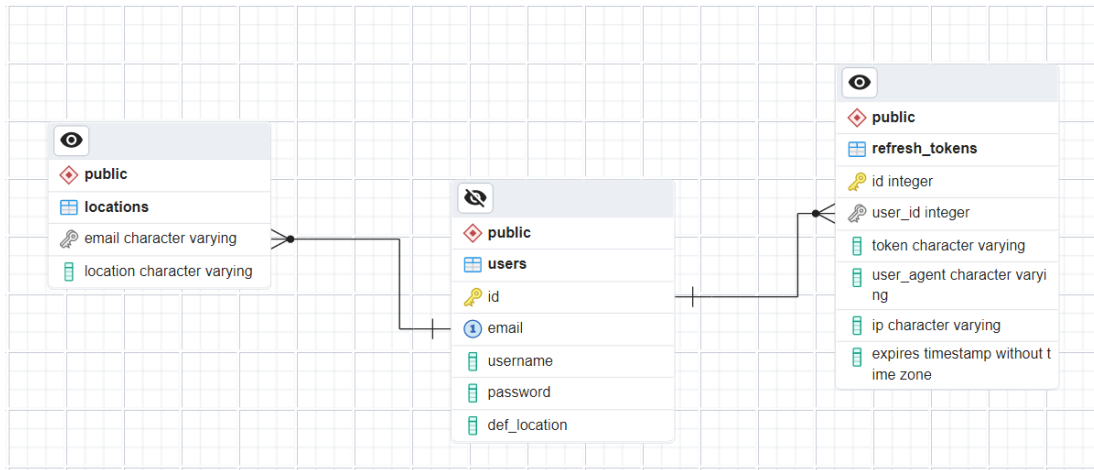
## 3.4. Baza podataka

Baza podataka implementirana je koristeći PostgreSQL, moćan sustav otvorenog koda (*engl. open-source*) za upravljanje bazama podataka koji omogućuje visoke performanse i skalabilnost. Baza podataka je dizajnirana da bude jednostavna i efikasna s

---

<sup>1</sup>Preuzeto sa <https://www.weatherapi.com/about.aspx>

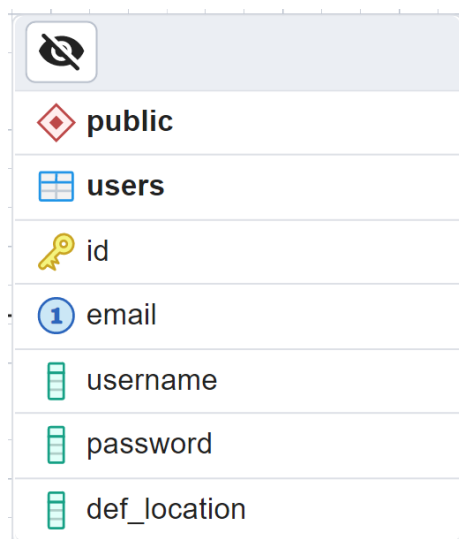
obzirom na specifične potrebe naše aplikacije. Sastoji se od tri glavne tablice: **korisnici** (engl. *users*), **lokacije** (engl. *locations*) i **osvježavajući tokeni** (engl. *refresh tokens*).



Slika 3.1: Shema baze podataka.

### 3.4.1. Tablica Users

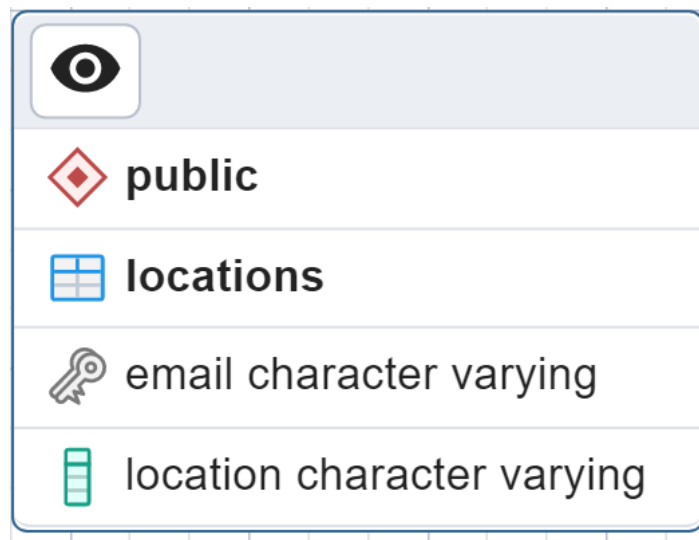
Tablica *users* sadrži informacije o registriranim korisnicima u našem sustavu. Svaki korisnik ima jedinstveni identifikator (ID), korisničko ime, lozinku i e-mail adresu koja je također jedinstvena.







Slika 3.2: Tablica *users*.

### 3.4.2. Tablica Locations

Tablica *locations* pohranjuje informacije o različitim lokacijama koje su korisnici unijeli kao svoje favorite. Sastoji se od e-maila i imena lokacije.



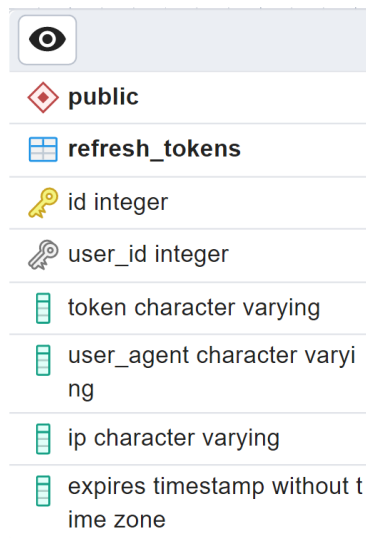
The diagram illustrates the structure of the *locations* table. It is represented as a vertical stack of five rows, each with a distinct icon and a label. The first row features an eye icon and the label 'public'. The second row has a grid icon and the label 'locations'. The third row displays a key icon and the label 'email character varying'. The fourth row shows a document icon and the label 'location character varying'. The entire structure is enclosed in a rounded rectangle with a blue border.

|   |                            |
|---|----------------------------|
|  | public                     |
|  | locations                  |
|  | email character varying    |
|  | location character varying |

Slika 3.3: Tablica *locations*.

### 3.4.3. Tablica Refresh Tokens

Tablica *refresh tokens* koristi se za pohranjivanje svih trenutno važećih osvježavajućih tokena za korisnike. Svaki zapis u tablici sadrži jedinstveni identifikator tokena, ID korisnika kojem token pripada, vrijeme isteka tokena, IP adresu i korisničkog posrednika (*engl. User-Agent*).



|  | public                              |
|--|-------------------------------------|
|  | refresh_tokens                      |
|  | id integer                          |
|  | user_id integer                     |
|  | token character varying             |
|  | user_agent character varying        |
|  | ip character varying                |
|  | expires timestamp without time zone |

Slika 3.4: Tablica *refresh tokens*.

## 3.5. Testiranje i ispravljanje pogrešaka

Testiranje i ispravljanje pogrešaka ključni su aspekti razvoja svake aplikacije. U ovom projektu koristili smo različite alate i pristupe kako bismo osigurali ispravnost našeg sustava.

### 3.5.1. Postman

Za testiranje komunikacije između klijenta i poslužitelja, kao i komunikacije poslužitelja s WeatherAPI sučeljem, koristili smo aplikaciju Postman [3]. Postman je popularan alat koji programerima omogućuje da kreiraju, dijele, testiraju i dokumentiraju aplikacijska sučelja. Svojim intuitivnim korisničkim sučeljem, Postman omogućava lako slanje zahtjeva prema aplikacijskim sučeljima i analizu odgovora. Također, omogućava grupiranje testova u kolekcije za jednostavnije testiranje.

### 3.5.2. VS Code i evidentiranje grešaka

Za razvijanje koda i njegovo praćenje koristili smo integrirano razvojno okruženje (*engl. IDE - Integrated Development Environment*) Visual Studio Code [6]. VS Code je visoko konfigurabilno okruženje koje podržava velik broj programskih jezika i alata te pruža korisne značajke poput naglašavanja sintakse, automatskog dovršavanja koda i dubinskog praćenja koda.

Važan dio ispravljanja pogrešaka bilo je i zapisivanje poruka prilikom prijenosa. Sustav je konfiguriran da evidentira ključne informacije tijekom izvođenja. Evidencija se

koristila za praćenje tijeka izvođenja, detektiranje i ispravljanje pogrešaka te za analizu performansi sustava. Korištenje ispisa zapisnika (*engl. logs*) se pokazalo ključnim u razvoju budući da je omogućilo brzo identificiranje i rješavanje problema.

## 4. Zaključak

Razvoj web aplikacije za pronalaženje vremenskih uvjeta zahtijevao je uspješnu integraciju više tehnologija i pristupa uključujući Flask, PostgreSQL, korisnički autentifikacijski sustav i sustav komunikacije s WeatherAPI-jem. Usmjerenost na jasnu strukturu koda i upotrebu modernih alata i praksi omogućilo je razvoj čistog, efikasnog i sigurnog sustava.

Kroz proces izgradnje aplikacije, posebno je naglašen značaj testiranja i ispravljanja pogrešaka. Korištenje aplikacije Postman bilo je instrumentalno za testiranje komunikacije između klijenta i poslužitelja, kao i između poslužitelja i API-ja. Isto tako, korištenje evidentiranja poruka i razvojnog okruženja VS Code omogućilo je efikasno praćenje i rješavanje potencijalnih problema u radu aplikacije.

Ukupno, ovaj projekt je pokazao kako se kombiniranjem različitih tehnologija i pristupa može stvoriti snažna i funkcionalna web aplikacija. Kroz daljnje poboljšavanje i dodavanje novih značajki, ova aplikacija ima potencijal da postane još korisniji alat za pružanje preciznih vremenskih prognoza.

# LITERATURA

- [1] *Flask*. Flask. URL: <https://flask.palletsprojects.com/en/2.3.x/>.
- [2] PostgreSQL. PostgreSQL. URL: <https://www.postgresql.org>.
- [3] *Postman*. Postman. URL: <https://www.postman.com/>.
- [4] *React Router*. React Router. URL: <https://reactrouter.com/en/main>.
- [5] *TypeScript and React*. URL: <https://www.typescriptlang.org/docs/handbook/react.html>.
- [6] *Visual Studio Code*. Microsoft. URL: <https://code.visualstudio.com/>.
- [7] WeatherAPI. URL: <https://www.weatherapi.com/>.



## **Sustav za praćenje vremenske prognoze**

### **Sažetak**

U okviru završnog rada je razvijen sustav za praćenje vremenske prognoze. Korišteni su Python i React Typescript, uz PostgreSQL bazu podataka. Implementirano je korisničko sučelje koje omogućuje interaktivan prikaz vremenskih podataka. Testiranje je provedeno korištenjem alata poput Postmana, logova i Visual Studio Code. Ostvarena je uspješna funkcionalnost sustava, što naglašava važnost korištenja različitih tehnologija i testiranja u razvoju web aplikacija.

**Ključne riječi:** vremenska prognoza, Python, React Typescript, PostgreSQL, korisničko sučelje, testiranje, Postman, web aplikacije

## **Weather forecast monitoring system**

### **Abstract**

As part of the final project, a weather forecast monitoring system has been developed. Python and React Typescript were used, along with PostgreSQL as the database. A user interface has been implemented to provide an interactive display of weather data. Testing was performed using tools such as Postman, logs, and Visual Studio Code. The system has achieved successful functionality, highlighting the importance of utilizing different technologies and testing in web application development.

**Keywords:** weather forecast, Python, React Typescript, PostgreSQL, user interface, testing, Postman, web applications