iPets

Projekt pri predmetu Spletne Tehnologije

Alen Fridau, Tibor Vito Šušnjara, Lea Tajnikar

Maribor, junij 2025

**1 Namen aplikacije**

Aplikacija iPets uporabnikom omogoča enotno mesto za podatke o svojih živalih na pregledni način. Uporabniki lahko dodajo svoje živali in osnovne podatke o njih, nato pa vnašajo podatke o hranjenju (datum, tip hrane, katera hrana je živali najljubša), veterinarskih obiskih (kraj in datum, opomniki, zapiski rezultatov po pregledu) in zdravljenju (zdravila, operacije, cepiva, itd.).

**2 Implementirane funkcionalnosti**

1. Upravljanje z živalmi
   * Dodajanje živali (ime, pasma, vrsta, rojstvo, slika)
   * Urejanje in brisanje živali
2. Uporabniški sistem
   * Registracija uporabnikov
   * Avtentikacija uporabnikov po standardu O Auth2.0
   * Osveževanje seje z “refresh tokens”
3. Upravljanje s hrano in hranjenjem
   * Dodajanje hrane (ime, tip)
   * Urejanje in brisanje hrane
   * Dodajanje hranjenja za posamezno žival (hrana, čas in ura)
   * Urejanje in brisanje hranjenja
4. Upravljanje z zdravili in zdravljenjem
   * Dodajanje zdravil (naziv, opis)
   * Urejanje in brisanje zdravil
   * Dodajanje zdravljenja (zdravilo, tip, doziranje, opis)
   * Urejanje in brisanje zdravljenja
5. Upravljanje z veterinarskimi obiski
   * Dodajanje obiskov (datum, čas, lokacija, opis)
   * Urejanje in brisanje obiskov
6. Dodatne funkcionalnosti
   * Omejeno delovanje brez povezave
   * Obvestila za prihajajoče veterinarske obiske
   * Bližnjice s tipkovnico
   * Upravljanje z mikrofonom

**2.1 Odgovornosti članov**

**Alen Fridau**: implementacija funkcionalnosti upravljanja s hrano in hranjenjem živali, postavitev in izgled »glave« strani ter strani za upravljanje s hrano

**Tibor Vito Šušnjara**: priprava osnovne strukture in skupnih komponent aplikacije, implementacija funkcionalnosti osnovnega upravljanja z živalmi, implementacija uporabniškega sistema in avtorizacije, postavitev in izgled seznama živali ter podrobnih podatkov o živali

**Lea Tajnikar**: implementacija funkcionalnosti upravljanja z zdravili in zdravljenjem živali, implementacija funkcionalnosti upravljanja z veterinarskimi obiski, implementacija »service worker« in notifikacij, implementacija bližnjic in upravljanja z mikrofonom, postavitev in izgled strani za upravljanje z zdravili in veterinarskimi obiski

**3 Implementacija**

**3.1 Zaledni del**

Zaledni del je implementiran z JavaScriptom v **Node.js** okolju. Uporablja ogrodje **Express**, za povezavo s podatkovno bazo pa knjižnico **knex**. Sama podatkovna baza je **SQLite**, ki se nahaja kot datoteka znotraj delovnega imenika aplikacije.

Express strežnik ima sledeče skupine končnih točk: users, pets, visits, treatments, medications, food in auth. Večinoma so končne točke znotraj posamezne skupine klasične CRUD operacije, npr. POST /users se uporabi za registracijo, GET /pets/user/:id pridobi vse žival uporabnika, ipd. Končne točke v »auth« skupini se uporabljajo za implementacijo avtorizacije po OAuth 2.0 standardu, in sicer so:

* POST /auth/token – pri veljavnem identifikatorju in geslu se vrneta žeton za dostop (»access token«) in osvežitev seje (»refresh token«)
* POST /auth/refresh – pri veljavnem žetonu za osvežitev se vrne novi žeton za dostop z istimi dovoljenji
* POST /auth/logout – pri veljavnem identifikatorju in geslu se morebitni shranjeni žetoni za osvežitev (za tega uporabnika) razveljavijo z izbrisom iz podatkovne baze

Druge uporabljene knjižnice v zalednem delu:

* »morgan« za izpisovanje podatkov o prihajajočih zahtevah v konzolo
* »dotenv« za nalaganje okoljskih spremenljivk iz datoteke (skrivnost za podpisovanje žetonov)
* »cors« za urejanje politike CORS, da omogoči dostop do strežnika iz vseh domen
* »bcrypt« za šifriranje in preverjanje gesel
* »jsonwebtoken« za podpisovanje in validacijo žetonov

**3.2 Čelni del**

Čelni del je implementiran z HTML, CSS in JavaScript z uporabo ogrodja **React**. Za pomoč pri razvoju in gradnji aplikacije smo uporabili orodje **vite** in okolje **Node.js**. Za navigacijo po aplikaciji, ki je zasnovana kot SPA (»single page app«), smo uporabili **React Router**. Upravljanje s pomočjo mikrofona je podprto s knjižnico **react-speech-recognition**.

Čelni del je implementiran kot progresivna spletna aplikacija – ima »service worker«, ki s pomočjo predpomnjenja skrbi za delovanje brez povezave, ter datoteko manifest.json z osnovnimi podatki o aplikaciji – to omogoča namestitev aplikacije neposredno na napravo, če brskalnik to podpira. Uporablja tudi obvestila operacijskega sistema. Za pogoste strani v aplikaciji so implementirane bližnjice z tipkovnico.

Druge uporabljene knjižnice v čelnem delu:

* »bootstrap« za lažje definiranje izgleda aplikacije
* »jwt-decode« za branje vsebine žetonov (brez validacije podpisa)

**4 Težave in reševanje**

Težav med razvojem ni bilo veliko. Omembe vredna je bila posledica delovanja predpomnilnika s pomočjo »service worker.« Ta namreč v naši aplikaciji uporablja »cache-first« strategijo, kar je povzročilo težave pri sprotnem testiranju ob razvoju novih funkcionalnosti. Ob posodobitvi izvorne kode kakšne React komponente je brskalnik namesto posodobljene komponente pridobil staro iz predpomnilnika, kar je vodilo v nepričakovano obnašanje in nelogične posledice določenih sprememb v kodi. Po ugotovitvi vzroka problema smo ga rešili preprosto z začasnim izklopom predpomnjenja med samim razvojem.

**5 Testiranje**

Rešitev smo testirali pogosto, sproti ob samem razvoju. Po končani implementaciji smo v celoti testirali aplikacijo z uporabo vseh funkcionalnosti. Avtomatiziranih testov zaradi časovnih omejitev nismo implementirali, tudi obseg funkcionalnosti ni prerasel mere, kjer bi ročno testiranje bilo zelo zamudno.

Aplikacijo smo največ testirali v brskalniku, najpogosteje v Google Chrome, saj podpira uporabljeno knjižnico za prepoznavo glasu ter namestitev progresivne spletne aplikacije brez dodatnih razširitev (za razliko od npr. Firefox). Testirali smo jo tudi v obliki, nameščeni na osebni računalnik.

S sprotnim testiranjem smo odkrili in odpravili veliko manjših napak, v končnem testiranju pa so bile odkrite le nekatere manjše napake v postavitvi oz. spregledani elementi, ki niso imeli določenega izgleda.